MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM+

IAN GRAHAM

LIBRO UNO

Imprescindible para el usuario
La primera guía
a todo color para la
programación

MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM+

LA SERIE «CURSO VISUAL» EN PANTALLA SOBRE PROGRAMACION

Hoy día hay una urgente necesidad de guías prácticas, bien hechas y sencillas, para aprender a usar un ordenador. Por eso se han creado las series sobre programación «Curso Visual en Pantalla». Son un concepto totalmente nuevo en el campo de la autoenseñanza de microinformática y constituyen la primera biblioteca de manuales de Programación Paso-a-Paso, altamente ilustrados, para máquinas específicas.

LIBROS SOBRE EL ZX SPECTRUM +

Este es el Libro Uno de la excepcional serie de guías «Paso-a-Paso» para programar el ZX Spectrum+. Junto con los otros volúmenes, constituye un curso de autoenseñanza que comienza con los principios básicos de programación y llega a alcanzar, a través de técnicas y rutinas más complejas, un nivel avanzado.

SOBRE OTROS ORDENADORES

Los otros volúmenes de la serie abarcarán los ordenadores más populares en el mundo. Estos incluirán:

Programación Paso-a-Paso para el **ZX Spectrum**

Programación Paso-a-Paso para el **Apple IIe**

Programación Paso-a-Paso para el **Commodore 64**

IAN GRAHAM

Después de licenciarse en Física Aplicada y obtener un diploma de Periodismo para graduados en la City University, Londres, Ian Graham ha trabajado como editor auxiliar de Electronics Today International y editor suplente de Which Video? Desde 1982, en que llegó a ser un escritor independiente de plena dedicación, ha colaborado en una amplia gama de revistas técnicas incluyendo Computing Today, Video Today, Video Search, Hobby Electronics, Electronic Insight, Popular HI-FI, Science Now, Next y otras, y también ha escrito varios libros muy conocidos sobre ordenadores e informática. Entre estos están Computer & Video Games, Information Technology, The Inside Story-Computers.

LIBRO UNO


sinclair
ZX Spectrum +
TRUE VIDEO
INV VIDEO
BLUE DEF FN
RED FN
MGNTA LINE
GREEN OPEN#
CYAN CLOSE#
YELLW MOVE
WHITE ERASE
POINT
CAT
BLACK FORMAT
DELETE
GRAPH
SIN ASN
PLOT
COS ACS
DRAW
TAN ATN
REM
INT VERIFY
RUN
RND MERGE
RAND
STR$
RETURN AND
CHR$
IF OR
CODE IN
INPUT AT
PEEK OUT
POKE
TAB
PRINT
EXTEND MODE
EDIT
READ
NEW STOP
RESTR
SAVE NOT
DATA
DIM STEP
SGN
FOR TO
ABS
GOTO THEN
SQR CIRCLE
GOSUB
VAL VAL$
LOAD
LEN SCRN$
LIST
USR ATTR
LET
PI INVERS
PAUSE
CAPS SHIFT
CAPS LOCK
LN BEEP
COPY
EXP INK
CLEAR
LPRNT PAPER
CONT
LLIST FLASH
CLS
BIN BRIGHT
BORDER
INKY$ OVER
NEXT
SYMBOL SHIFT

MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM+

IAN GRAHAM

MicroTextos

LIBRO UNO

SUMARIO

Una edición de
MicroTextos s. a. de ediciones
Paseo de la Castellana 166
28046 Madrid
Teléf. 458 65 91 Telex 49416

Editora Teresa Santiago
Traducción Tratecsa

Editor del proyecto David Burnie
Editor artístico Peter Luff
Diseñador Steve Wilson
Fotografía Vincent Oliver
Director de edición Alan Buckingham
Director artístico Stuar Jackman

Obra original Step-by-Step Programming ZX Spectrum+
Publicada por Dorling Kindersley Ltd., London

ISBN: Obra completa; 84-86376-01-7
ISBN: 84-86376-03-3
Depósito Legal M-16.106-1985

Fotocomposición
Pérez-Díaz, S. A., Madrid

Fotomecánica
Progreso Gráfico, S. A., Madrid

Impresión y Encuadernación
Unigraf, S. A., Madrid

EL ZX SPECTRUM +

El ZX Spectrum + es una versión mejorada del microordenador ZX Spectrum, que desde su lanzamiento en 1982, se ha convertido en el ordenador personal más popular. El ZX Spectrum + contiene un circuito como el del ZX Spectrum, pero está alojado en una carcasa con un teclado muy mejorado que hace mucho más sencillo su manejo que el de su predecesor. Este nuevo ordenador tiene los mismos componentes básicos que otros ordenadores mayores, y usa su propia versión de BASIC (lenguaje de microordenadores más conocido). El ZX Spectrum + ofrece un modo barato de aprender a programar un ordenador con muchas de las características que poseen los ordenadores mayores y más complejos. Además, el ZX Spectrum+ tiene una memoria de 48K, memoria superior a muchas de las que poseen algunos de mayor precio.

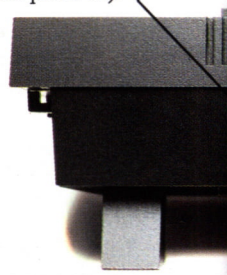
Conectores y periféricos

Por fuera, el Spectrum parece una caja negra con un teclado en la cara superior. Las teclas de letras y números están dispuestas como en una máquina de escribir pero actúan de forma muy distinta. Al usar las de alrededor, cada tecla puede producir palabras completas e incluso figuras y símbolos. De hecho, cada una puede realizar hasta seis funciones diferentes.

Si se da la vuelta al ordenador y se inspecciona el panel trasero, se observan cuatro pequeños enchufes y una ranura. Los enchufes conectan el Spectrum a una televisión, a un magnetófono y a una fuente de alimentación. En la página 60 se pueden encontrar las instrucciones de uso del magnetófono con el ordenador. La ranura alargada es un conector, que es, de hecho, una parte del circuito del Spectrum. Las tiras metálicas del borde se utilizan para conectar otras piezas de *hardware*, tales como impresoras, microunidades o palancas de control, al ordenador. Se aconseja no tocar estos contactos puesto que tanto la grasa como la suciedad pueden provocar un mal funcionamiento.

El Spectrum produce imágenes de color, así como en blanco y negro. Además, produce todos los efectos sonoros utilizados en los programas. Si se mira el panel inferior de la máquina se puede ver un círculo de orificios en una esquina: esta es la salida del pequeño altavoz del ordenador. Es capaz de pitar o de reproducir melodías sencillas.

Enchufe a 9 voltios DC Adaptador de corriente que transforma corriente alterna (AC) en corriente directa (DC) a 9 voltios (adecuada para él).

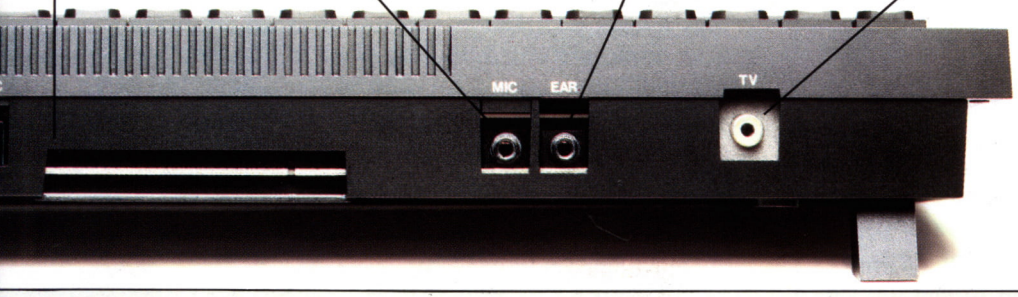


Conector Este enchufe conecta el ordenador a los periféricos, normalmente a través de *interface* ZX.

Enchufe MIC Permite transferir programas desde la memoria del ordenador a una cinta, al conectarlo al magnetófono.

Enchufe EAR Permite la recepción de programas almacenados en cinta. Se conecta al enchufe EAR del magnetófono.

Enchufe TV Las señales ópticas que el Spectrum produce se alimentan al televisor conectando el enchufe de la antena al del TV.



DENTRO DEL ORDENADOR

El ZX Spectrum+ está construido sobre una tarjeta de circuito. Sus componentes son circuitos integrados o *chips*; estos parecen pequeñas placas de plástico, provistas de hasta 20 puntas metálicas que sobresalen de cada extremo. Las puntas conectan los *chips* a los contactos situados en la superficie de la tarjeta.

El corazón del Spectrum es un microprocesador que hace el papel de Unidad Central de Proceso (CPU). La CPU realiza todos los cálculos del ordenador, supervisa el teclado y actúa cuando detecta la pulsación de una tecla. A pesar de su complejidad, la CPU sólo es capaz de ejecutar las instrucciones que se le dan.

Las instrucciones que se enuncian por medio del teclado deben ser traducidas en primera instancia al código máquina que el ordenador utiliza.

El programa que realiza la traducción reside permanentemente en Memoria de Sólo Lectura o "memoria no volátil" (ROM). En este tipo de memoria no se pueden almacenar los programas del usuario, ya que su contenido sólo puede ser leído. El hecho de encender o apagar el ordenador no afecta al programa traductor.

La memoria de Acceso Aleatorio (RAM) ofrece al usuario espacio para sus programas. Todos los programas permanecen en memoria RAM hasta que se introduce la palabra clave NEW o se aprieta el botón. Como el contenido de una RAM desaparece cuando se desconecta la alimentación, se denomina también "memoria volátil".

Principales componentes de la tarjeta

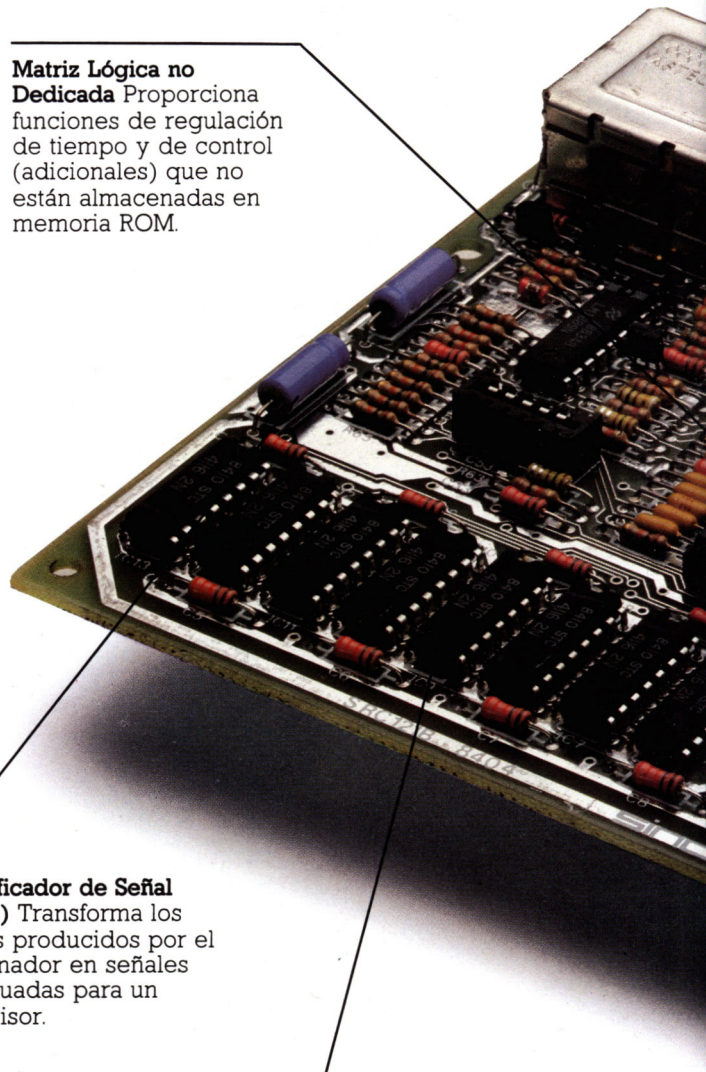
El ZX Spectrum+ tiene una memoria de 48K, es decir, puede almacenar 48 kilobytes de programas e información. Cada Kilobyte es igual a 1.024 bytes. Un Byte es una unidad estándar de información en forma binaria. Esta formado por 8 bits, que son pulsos eléctricos que representan un 1 ó un 0. Se puede pensar que los bytes son palabras y los bits letras. La diferencia fundamental entre el sistema de comunicación del Spectrum y el español es que todas las palabras que utiliza el ordenador son de 8 letras, formadas a partir de un alfabeto de sólo 2 letras distintas.

Las actividades del ordenador están sincronizadas, de forma que siempre esté disponible la información correcta en el lugar pertinente y en el momento adecuado. La sincronización la lleva a cabo un reloj interno, formado por un oscilador de cristal capaz de producir 3,5 millones de pulsos por segundo.

De otras operaciones adicionales de regulación de tiempo y de control se encarga un *chip* llamado matriz lógica no dedicada (ULA).

Cadena de órdenes de microchip Todos los *chips* del Spectrum forman una cadena de mandatos en la que la CPU realiza las tareas ejecutivas. El resto de los *chips* —incluyendo ROMS, RAMS y ULA— actúan como sistemas de almacenamiento temporal o permanente. Proporcionan a la CPU la instrucción que debe ejecutar.

En esta visión del interior del Spectrum, se han despegado los conectores que unen el teclado al resto del ordenador. Se aconseja no abrir el Spectrum ya que estos conectores se rompen fácilmente.



Matriz Lógica no Dedicada Proporciona funciones de regulación de tiempo y de control (adicionales) que no están almacenadas en memoria ROM.

Codificador de Señal (PAL) Transforma los datos producidos por el ordenador en señales adecuadas para un televisor.

Memoria de acceso aleatorio (RAM) Dieciséis *chips* de RAM en dos bloques de 8 *chips* suministran 48K, de almacenamiento para programas e información, al ordenador cuando está encendido.

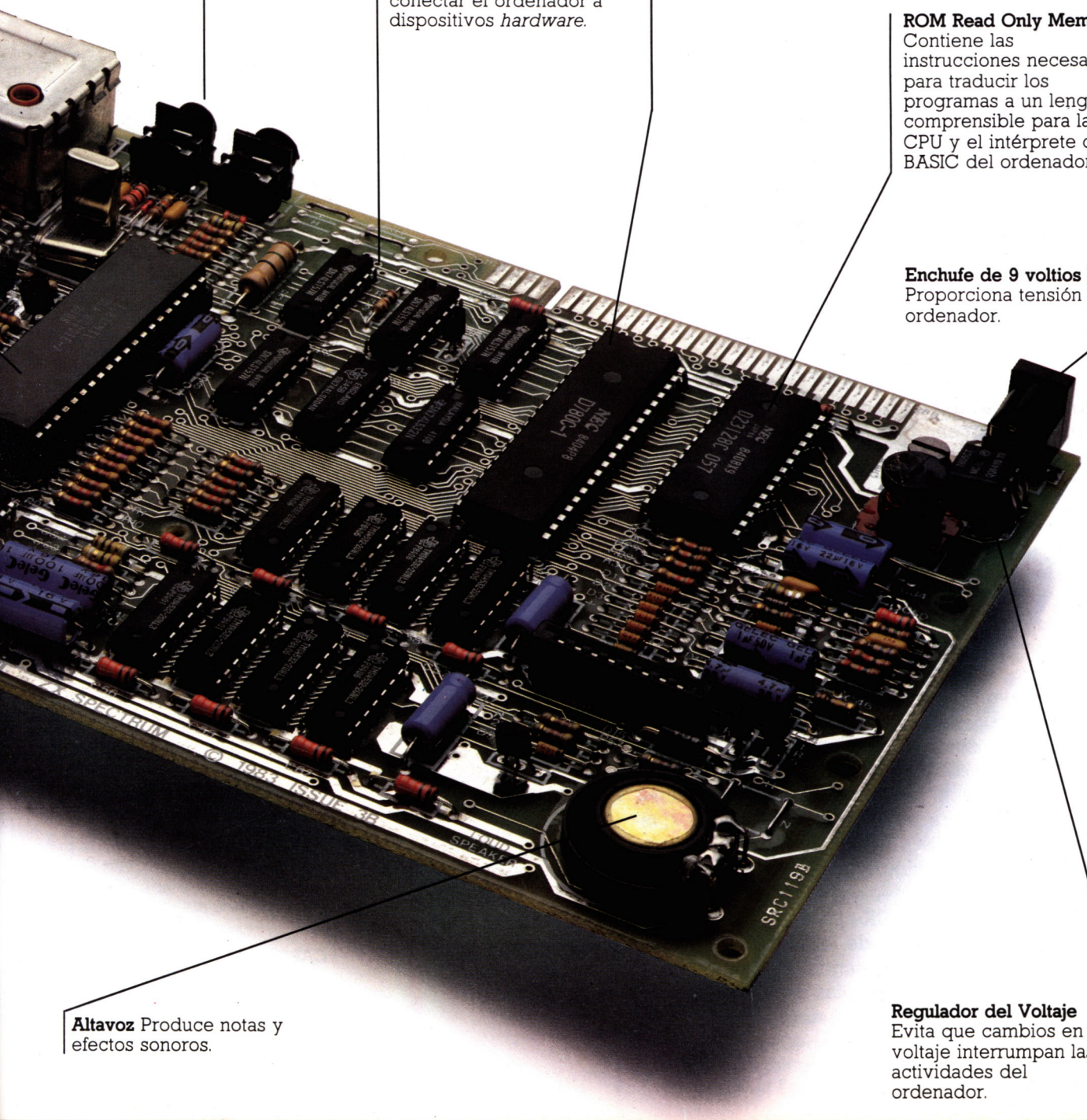
Enchufes de grabación Se utilizan para grabar programas.

Unidad Central de Proceso Es la parte inteligente del ordenador. Realiza todos los cálculos y actividades de control sobre el resto del sistema, a partir de información contenida en ROM y RAM.

Chips lógicos Gestionan funciones lógicas varias, como las necesarias al conectar el ordenador a dispositivos *hardware*.

ROM Read Only Memory Contiene las instrucciones necesarias para traducir los programas a un lenguaje comprensible para la CPU y el intérprete de BASIC del ordenador.

Enchufe de 9 voltios DC Proporciona tensión al ordenador.



Altavoz Produce notas y efectos sonoros.

Regulador del Voltaje Evita que cambios en el voltaje interrumpan las actividades del ordenador.

TECLADO DEL SPECTRUM

En un teclado de máquina de escribir convencional cada tecla se utiliza para imprimir una letra minúscula y con la tecla de mayúsculas, su versión en mayúsculas. El teclado del Spectrum funciona así; si bien es mucho más versátil, ya que posee más de una tecla de mayúsculas. Las teclas del Spectrum pueden seleccionar hasta seis funciones diferentes. Sus 58 teclas pueden producir un total aproximado de 200 letras, palabras y símbolos. En principio, el uso del teclado puede parecer arduo, pero, una vez que se dominan las teclas de mayúsculas se simplifica mucho la tarea: dos teclas de mayúsculas diferentes —CAPS SHIFT y SYMBOL SHIFT— se utilizan tanto por separado como juntas para seleccionar funciones.

La técnica del teclado

El Spectrum+ posee un teclado estándar, con teclas del tipo de recorrido total. Cada vez que se pulsa una tecla, produciendo la aparición de una letra o una palabra en la pantalla, el ordenador hace un ruido para hacerle saber que el contacto se ha realizado.

Si usted aprieta una tecla durante más de dos segundos, el signo se imprimirá repetidamente.

No hay que ser un gran mecanógrafo para usar el teclado, sobre todo porque las palabras que transmiten órdenes al ordenador con solo apretar una tecla aparecen en pantalla.

El cursor

El Spectrum le dice a Vd. qué tipo de instrucción o símbolo espera, mediante uno de los cinco cursores centelleantes que posee. Si el cursor es una "K", entonces el ordenador espera una orden. Presione una letra, y la orden que aparece en la parte superior de dicha tecla se produce; apretando la P, por ejemplo, se produce la orden PRINT. Si hay más de una clave en la parte superior de una tecla, entonces la de más arriba aparece en pantalla. Después de que una clave aparece en pantalla, el cursor pasa a "L", que significa que las teclas de letra producen ahora tan sólo letras, excepto que Vd. apriete a la vez SYMBOL SHIFT. Apretando la tecla P ahora se produce la p minúscula. Para que aparezca la P mayúscula, presione la tecla CAPS SHIFT a la vez. Apretando la tecla CAPS LOCK aparece el cursor "C", que hace que todas las teclas de letra produzcan mayúsculas. De forma análoga pulsando GRAPH aparece el cursor "G" que hace que las teclas del 1 al 8 produzcan caracteres gráficos. El quinto cursor es "E", que aparece con la tecla EXTEND MODE. El ordenador ahora produce cualquiera de las dos palabras claves que están encima de la parte superior de cada tecla.

GRAPH Esta tecla conecta el Spectrum al "modo gráfico", preparado así para producir caracteres gráficos con las teclas del 1 al 8 y caracteres gráficos definidos por el usuario con las teclas de la A a la U. Pulse GRAPH de nuevo para desconectar el modo gráfico.

NEW Esta tecla borra cualquier programa o dato de la memoria del ordenador.

DELETE Esta tecla hace retroceder el cursor y borra cualquier palabra clave, letra o símbolo de una sentencia o línea que se esté editando.

EDIT Se usa para sacar una línea de un programa y poderla editar.

EXTEND MODE Pulse esta tecla primero y después cualquier letra para seleccionar la palabra clave superior de las dos que aparecen en la parte alta de la tecla. B da BIN, por ejemplo. (con las teclas 1 al 7 Vd. conseguirá los códigos de color). Para que aparezca la clave de más abajo, utilice SYMBOL SHIFT.

CAPS SHIFT Esta tecla le permite a Vd. seleccionar la mayúscula de una letra.

CAPS LOCK Pulse esta tecla a la vez que cualquier otra para conseguir mayúsculas sin tener que usar CAPS SHIFT. Púlsela de nuevo para volver a las minúsculas.

TRUE VIDEO e INV VIDEO
Estas teclas producen color en textos o imágenes gráficas.

Teclas de imagen en color
Estas teclas determinan los colores que aparecen en la pantalla.

Teclas numéricas Ofrecen una forma fácil y rápida de producir gráficos cuando se usan después de haber pulsado GRAPH. El carácter gráfico representado en la parte superior, aparecerá entonces. La parte blanca del carácter es la que tiene color, y la parte negra mantiene el color del papel. Para invertir los colores presione CAPS SHIFT y la tecla numérica.

BREAK Esta tecla interrumpe una programa en ejecución, pero no lo borra de la memoria.

ENTER El Spectrum no responderá a la mayoría de las órdenes o información tecleada a menos que vaya seguida de esta tecla. Cuando se pulsa esta tecla *ENTER*, el Spectrum responderá a las órdenes o almacenará la información en la memoria.

SYMBOL SHIFT Pulse esta tecla para utilizar la palabra clave de más abajo de la parte alta de la tecla. SYMBOL SHIFT y B pulsadas a la vez dan*; por ejemplo. Con el "extended mode" (cursor E), SYMBOL SHIFT y una letra o tecla numérica produce la palabra clave más baja o el símbolo que está encima de la parte alta de la tecla; B ahora da BRIGHT.

Espaciador Esta barra produce un espacio.

BEEP Se usa pulsando SYMBOL SHIFT y EXTEND MODE a la vez. Controla el sintetizador de sonido del Spectrum.

Control del cursor Estas cuatro teclas hacen moverse al cursor en el sentido de la flecha.

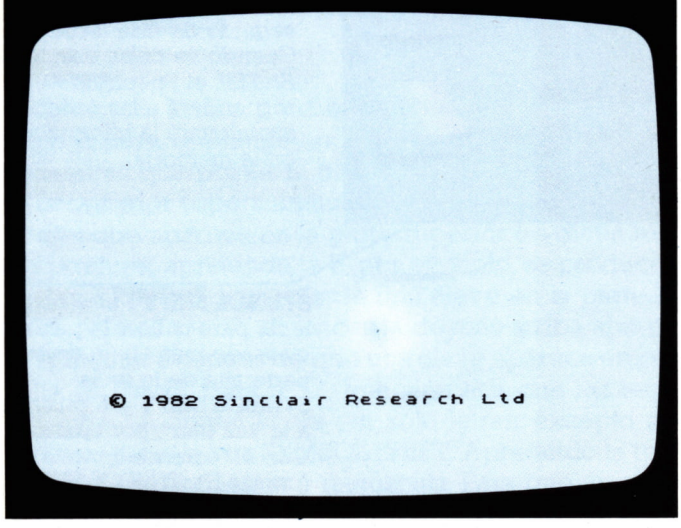
LA PUESTA EN MARCHA

La instalación del Spectrum es directa y lógica. Pero conseguir los mejores resultados posibles requiere algo más de tiempo. Para empezar, se conecta la fuente de alimentación del Spectrum a la alimentación general y, a continuación, al ordenador. El Spectrum no tiene un conmutador de encendido/apagado. Tan pronto como se conecta la fuente, si se ha realizado la conexión, se oye un zumbido.

A continuación hay que conectar el ordenador a la televisión. Con este fin, se conecta un extremo del cable negro, suministrado con el ordenador, a la toma de antena del televisor; el otro extremo se introduce en el enchufe marcado "TV" del Spectrum, y se enciende el televisor.

Posiblemente, al principio no se verá ningún tipo de imagen nítida en la pantalla y se percibirá un intenso siseo procedente del televisor. Debe disminuirse el volumen del televisor hasta el mínimo, puesto que el Spectrum produce todos los efectos sonoros requeridos por los programas con su propio altavoz. Se selecciona una canal de televisión que pueda conectarse permanentemente al ordenador. El televisor trata la señal emitida por el Spectrum como una transmisión habitual; esto obliga a sintonizar el televisor como se haría para ver un programa cualquiera. Se ajustan los controles hasta que la pantalla se configure así:

SEÑAL DEL SPECTRUM EN LA PANTALLA



Si no se obtiene ninguna imagen del ordenador compruebe que todas las conexiones son correctas, que el enchufe "TV" está conectado a la toma de antena del televisor y cerciórese de que el canal que se está sintonizando es el seleccionado. Sólo debería sintonizarse el televisor una vez.

Nada más sintonizar el canal adecuado, aparecerá la imagen correctamente.

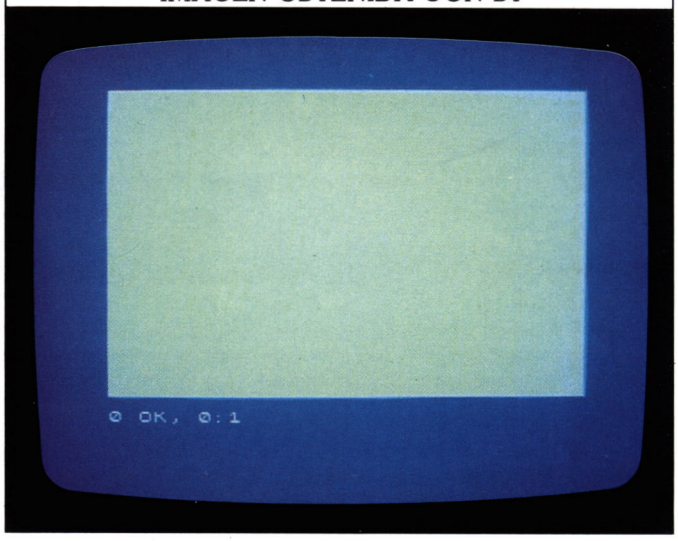
Cómo probar los colores del Spectrum

Para poder probar los colores del Spectrum se puede utilizar la siguiente secuencia de órdenes: (ENTER indica la tecla titulada ENTER que está a la derecha en el teclado):

```
B1 ENTER
B2 ENTER
B3 ENTER
B4 ENTER
B5 ENTER
B6 ENTER
B7 ENTER
```

Cada vez que se acciona la tecla ENTER, debe apreciarse un cambio de color en el borde de la pantalla. (color en páginas 34-35). B1 ENTER debe colocar la pantalla así:

IMAGEN OBTENIDA CON B1



Puede darse el caso de que esta secuencia de órdenes no produzca ningún efecto. La sintonización necesaria para tomar la señal de color del Spectrum adecuadamente es muy delicada; por esto hay que realizar repetidas pruebas hasta conseguir buenos resultados. Si se consigue una imagen pero no se obtiene color en la pantalla del televisor, puede darse el caso de que el televisor no interprete las señales de color del Spectrum. El proveedor le aconsejará en este caso.

A pesar de que la salida del Spectrum es habitualmente visible en un receptor de televisión ordinario, la señal puede ser alimentada en otro tipo de televisor llamado "monitor". El monitor contiene los mismos dispositivos que un televisor normal con excepción del sintonizador. Al transformar el flujo de datos del Spectrum en señal de te-

levisión de alta frecuencia y luego invertir el proceso en el televisor, se reduce la calidad de imagen. Eliminando estas etapas, el monitor produce imágenes de mayor calidad. Las fotografías de la pantalla de este libro se han realizado utilizando un monitor.

Conexión de periféricos

En la página 60 se explica detalladamente cómo utilizar una cinta de grabación para conservar los programas. Si se quiere utilizar el magnetófono, lo primero que hay que hacer es asegurarse de que se dispone del cable conector de color blanco y gris. Es muy importante hacer las conexiones correctamente o no se podrá utilizar el magnetófono.

Para conectar otros aparatos, debe Vd. utilizar el conector de la parte de atrás del ordenador. La impresora ZX de Sinclair se enchufa directamente a este conector. Para otros periféricos, como microunidades, palancas de juegos, etc., primero tiene que enchufar una *interface* ZX en el conector y después conectar el periférico a la *interface*.

No debe encajarse ningún enchufe en el conector si se observa la más mínima resistencia. Se podría estar introduciendo el enchufe incorrectamente y esto puede dañar el sistema.

Disposición del ordenador Con el fin de usar el ordenador de la forma más cómoda posible, la pantalla del televisor debe colocarse alineada detrás del ordenador. La pantalla no debe estar demasiado cerca. Las cintas grabadas con programas deben mantenerse alejadas de la fuente de alimentación, del televisor y del ordenador. De otra forma, los campos magnéticos generados por estos componentes del sistema podrían dañarlos.

USO DE LA PANTALLA

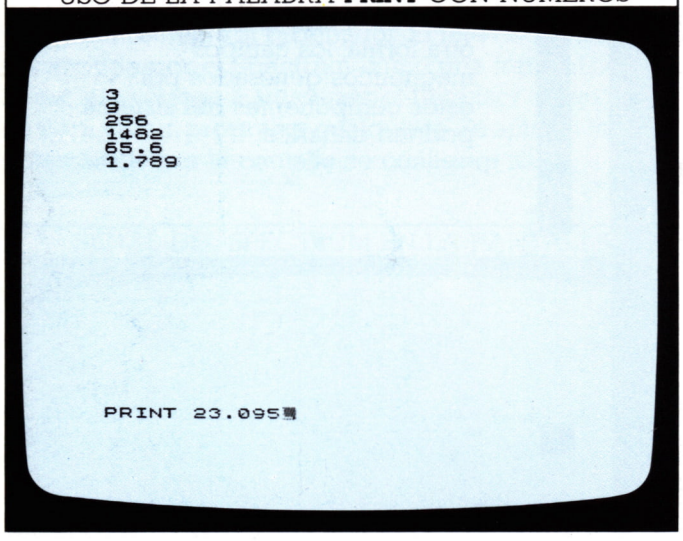
Con el ordenador ya instalado, seguro que ya ha caído en la tentación de teclear algo. Si todavía no lo ha hecho, inténtelo, no puede causar ningún daño. Lo primero que se advierte es que desaparece la línea escrita en la parte baja de la pantalla y que en su lugar aparece una línea de carácteres.

Pero como ya se ha indicado en las páginas 10-11, el que aparezca un carácter u otro depende de la combinación de teclas que se accione.

Comenzar a imprimir

Para dar algún sentido a esta situación que parece tan confusa presione la tecla RESET de la izquierda para borrar la memoria del ordenador. A continuación presione la tecla PRINT (tecla P) seguida de una tecla numérica y de la tecla ENTER. Tan pronto como se presione ENTER, el número elegido aparece en la parte superior de la pantalla. Se puede usar la palabra clave PRINT para imprimir una secuencia de números:

USO DE LA PALABRA **PRINT** CON NUMEROS



Si en vez de desconectar el ordenador se acciona la tecla marcada CLS (tecla V), seguida de ENTER, se borra la pantalla.

¿Qué es una variable?

Teclee:

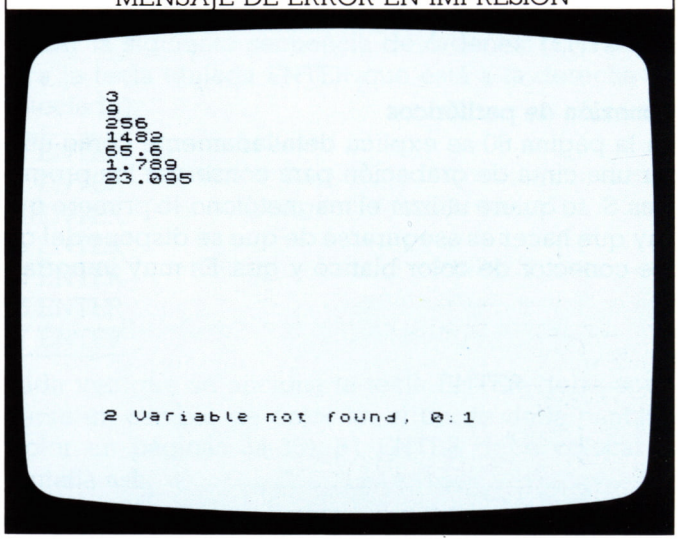
```
PRINT x
```

Y accione la tecla ENTER. El ordenador responde a esto —o a cualquier orden de imprimir una letra— con el siguiente mensaje de error:

```
2 Variable not found, 0:1
```

Este mensaje indica por qué no puede ejecutar la orden que se le ha dado:

MENSAJE DE ERROR EN IMPRESIÓN



En vez de escribir "X" en la pantalla, ha estado buscando una variable en la memoria.

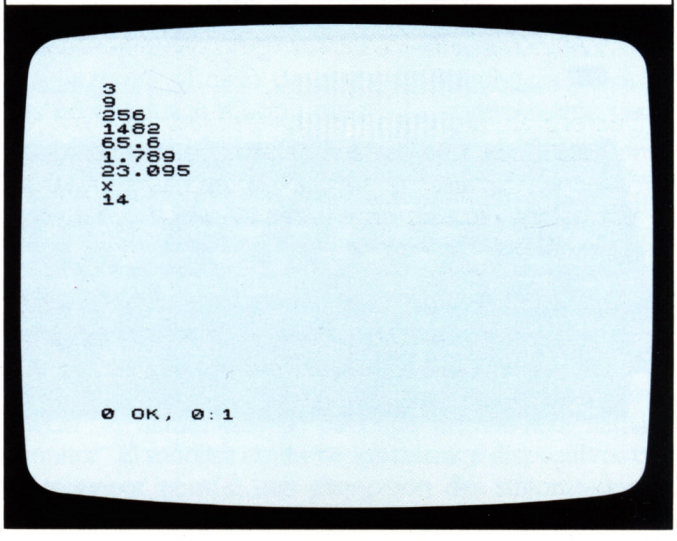
Use la tecla de las comillas, y teclee:

```
PRINT "x"
```

Al accionar ENTER, el ordenador imprime una x en la siguiente línea. Esto demuestra que para el ordenador x y "x" tienen distintos significados. El ordenador trata una letra sola como una variable. Una variable no es más que una etiqueta que identifica a un número almacenado en la memoria del sistema. Para que la orden PRINT x sea comprendida por el ordenador, hay que dar un valor a x (no olvide presionar la tecla ENTER después de cada línea):

```
LET x = 14
PRINT x
```

USO DE **LET** Y **PRINT**



Ahora x representa el número 14. LET (en la tecla L) es una orden que otorga un nombre y un valor a una posición de memoria. Como x representa siempre un número, se le denomina variable numérica.

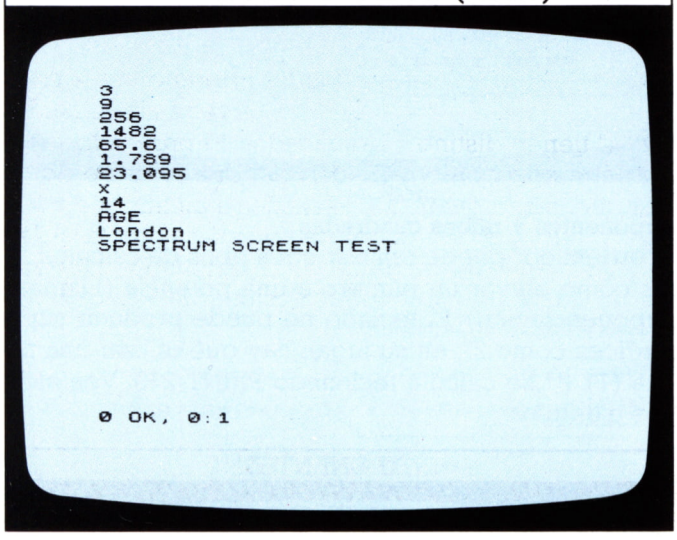
Cómo usar cadenas

Entonces x es una variable numérica pero "x" no lo es. El ordenador exhibe todos los símbolos encerrados entre comillas tal y como se teclean. Se puede usar de esta forma cualquier carácter del teclado —letras, números, símbolos matemáticos y de puntuación. Se sugiere teclear algunos ejemplos. Recuerde que la palabra clave PRINT puede usarse tantas veces como se desee si se acciona a continuación la tecla ENTER. Las mayúsculas se producen con las teclas CAPS SHIFT y CAPS LOCK:

```
PRINT "AGE"
PRINT "LONDON"
PRINT "SPECTRUM SCREEN TEST"
```

Los caracteres encerrados entre comillas dobles se denominan cadenas. De la misma forma que se almacena un número en el ordenador representado por una variable numérica, una cadena se almacena y se representa con una variable de tipo cadena. Las variables de tipo cadena son siempre letras y acaban con un símbolo de dólar $. En la línea:

IMPRIMIENDO CADENAS **(PRINT)**




```
LET A$="LONDON"
```

A$ es la variable de tipo cadena y LONDON es la cadena que la representa. Una vez escrita esta línea, borre la pantalla con CLS y teclee:

```
PRINT A$
```

Una vez accionada la tecla ENTER, el ordenador exhibe el contenido de la variable.

Impresión posicional con TAB y AT

Se prueba ahora una nueva modalidad con la palabra clave PRINT:

```
PRINT "Uno", "Dos", "Tres", "Cuatro", "Cinco", "Seis", "Siete", "Ocho".
```

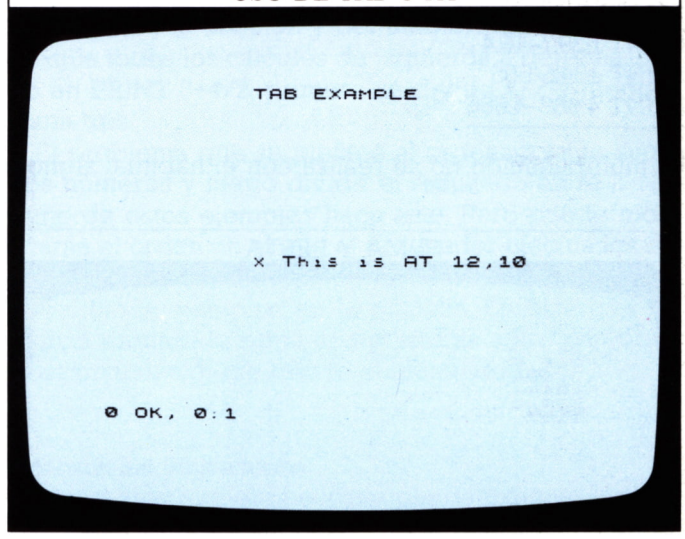
Al apretar la tecla ENTER aparecen estas palabras sobre la pantalla, en dos columnas. De hecho, la pantalla está divida en dos campos invisibles, cada uno de 16 caracteres de anchura. "Uno"se imprime en el primer campo, "Dos" en el segundo, "tres" de nuevo en el primero etc.

De todas formas, las palabras clave TAB (tecla P) y AT (tecla I), permiten imprimir caracteres o cadenas en cualquier lugar de la pantalla. Limpie la pantalla con CLS y teclee las dos líneas siguientes.

```
PRINT TAB 10; "TAB EXAMPLE"
PRINT AT 12,10; "x THIS is AT 12,10"
```

Si se acciona ENTER después de cada línea, la pantalla debe adquirir el siguiente aspecto:

USO DE **TAB** Y **AT**



TAB funciona de la misma forma que el tabulador de una máquina de escribir. El número que sigue a la palabra clave indica la posición del carácter. Hay 32 posiciones en una línea —las posiciones para la orden TAB están numeradas del 0 al 31 empezando por la izquierda.

AT es similar a TAB, pero además permite especificar la posición vertical en una línea.

AT se usa siempre con una pareja de números separados por una coma. El primero indica el número de la línea, contando desde la parte superior hacia la inferior de la pantalla. Hay 22 líneas en la pantalla, numeradas de 0 a 21. El segundo número indica la columna y es idéntico al que acompaña a la palabra clave TAB. La orden AT debe llevar siempre una coma, si no el ordenador no lo comprende.

CALCULOS CON EL ORDENADOR

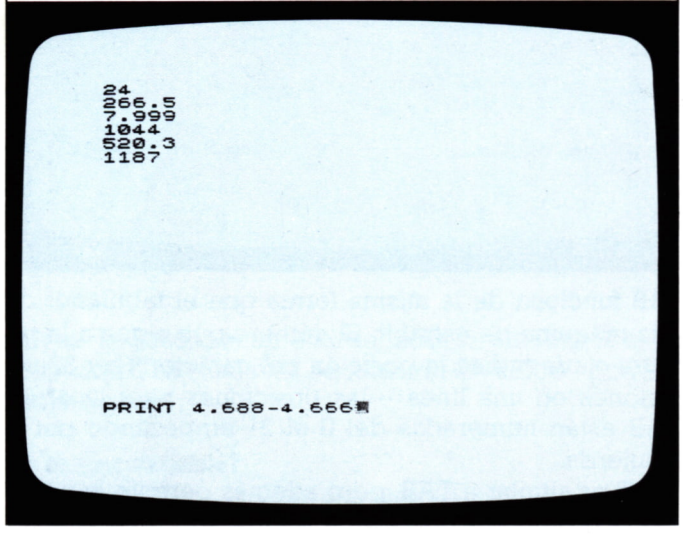
La palabra clave PRINT no está limitada a mostrar caracteres en la pantalla. Puede usarse junto con las cuatro operaciones matemáticas: suma, resta, multiplicación y división para realizar cálculos sobre la pantalla del televisor.

En lo que a la suma respecta, el signo más está en la tecla K. Como se trata de un símbolo de más abajo de la parte alta de la tecla, pulse SYMBOL SHIFT al mismo tiempo que K para escoger el carácter requerido. La resta se lleva a cabo de forma análoga. El signo menos, que hace el papel del guión cuando se utiliza en un texto, está situado en la tecla J. Al igual que el signo más, se obtiene con la tecla SYMBOL SHIFT. Aquí tiene algunos ejemplos, junto con los resultados que producen. Recuerde pulsar ENTER al final de cada línea para que el ordenador realice el cálculo.

```
PRINT 6+18
PRINT 250+16.5
PRINT 1.999+6
PRINT 905+139
PRINT 539.7-19.4
PRINT 1842-655
PRINT 4.688-4.666
```

La multiplicación no se realiza con el habitual signo ×

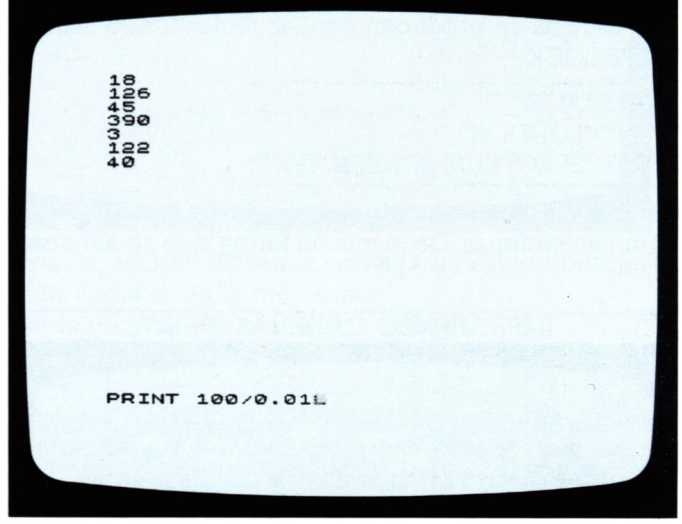
SUMA Y RESTA



sino con una estrella o asterisco *. El asterisco es el carácter SYMBOL SHIFT rojo situado en la tecla B, en la última fila del teclado. La división utiliza el trazo oblicuo / que es un carácter del tipo SYMBOL SHIFT situado en la tecla V. En la expresión 24/8 el número de la izquierda se divide entre el de la derecha. Para computar el primero de los siguientes ejemplos, teclee PRINT 3*6 y a continuación ENTER.

```
PRINT 3*6
PRINT 14*9
PRINT 2.5*18
PRINT 5*78
PRINT 24/8
PRINT 366/3
PRINT 600/15
PRINT 100/0.01
```

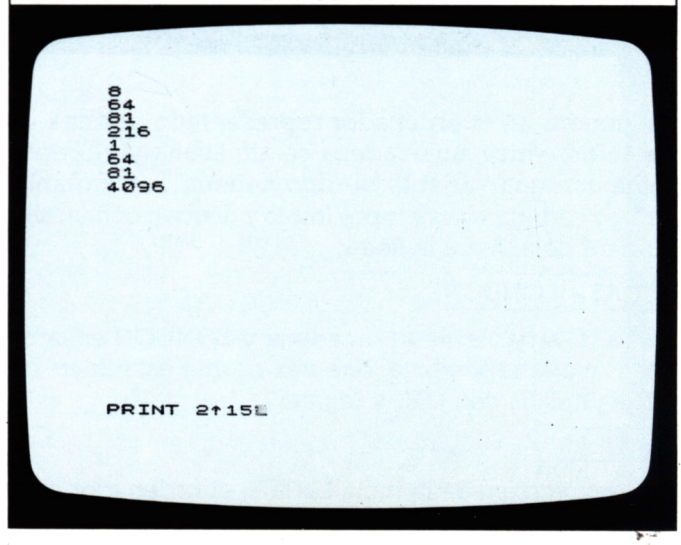
MULTIPLICACION Y DIVISION



Exponentes y raíces cuadradas

El ordenador puede realizar otros tipos de cálculos, tales como elevar un número a una potencia (Llamado exponenciación). El teclado no puede producir superíndices como 2^3; en su lugar, hay que utilizar una flecha (↑). 2^3 se calcula tecleando PRINT 2↑3. Vea algunos ejemplos:

EXPONENTES



El ordenador permite también calcular la raíz cuadrada de un número. La orden es SQR, la función EXTEND MODE en la tecla H.

```
PRINT SQR 2
```

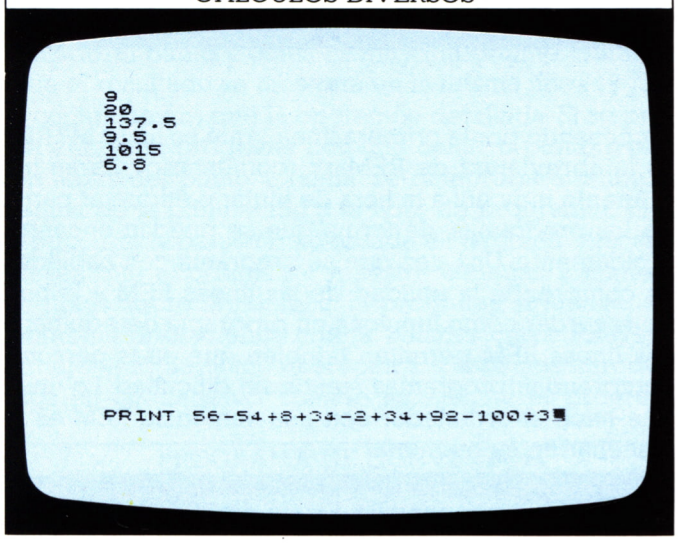
Cuando, una vez escrita esta línea, se introduce ENTER, el ordenador imprime el resultado. Pero si se prueba esta operación con un número negativo, el ordenador responde con un mensaje de error para notificar que se ha solicitado una imposibilidad matemática.

Conseguir el orden correcto

Se pueden realizar simultáneamente cálculos diversos con un único PRINT. Probemos primero con suma y resta:

```
PRINT 2+6+3+7-8+3-4
PRINT 48-42+16-2
PRINT 122-19+32+2.5
PRINT 4.8+2.8+1.9
PRINT 1024+14-23
PRINT 15.5-12.5+7.6-3.8
PRINT 56-54+8+34-2+34+92-100+3
```

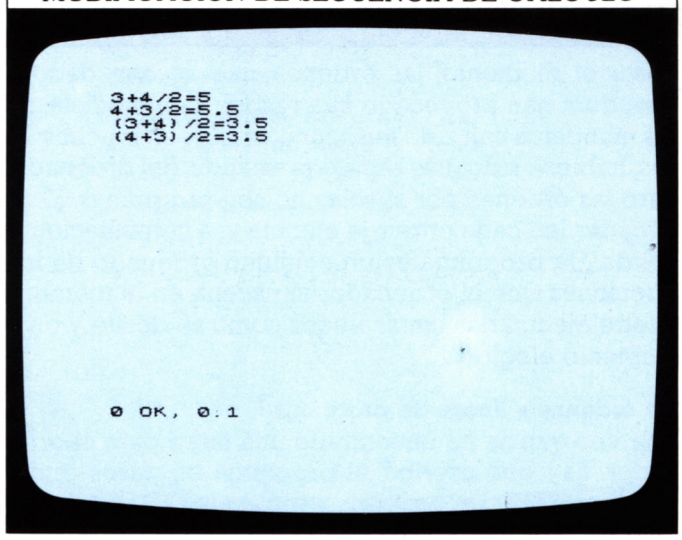
CALCULOS DIVERSOS



Independientemente del orden en el que se introduzcan los números, se obtiene siempre el mismo resultado. De todas formas, si se añaden multiplicaciones y divisiones a la secuencia de operaciones, pueden obtenerse resultados aparentemente extraños. Supongamos que se desea sumar dos números y dividir el resultado entre 2. El orden en el que se introducen los números no debería afectar al resultado pero, aparentemente, sí lo hace. Cada una de las siguientes líneas imprime una operación y un resultado:

```
PRINT "3+4/2="; 3+4/2
PRINT "4+3/2="; 4+3/2
PRINT "(3+4)/2="; (3+4)/2
PRINT "(4+3)/2="; (4+3)/2
```

MODIFICACION DE SECUENCIA DE CALCULO



Si 3+4 es lo mismo que 4+3, ¿Por qué la división entre dos introduce alguna diferencia? El motivo es que el ordenador no tiene por qué realizar los cálculos en el mismo orden en el que se escriben en la pantalla. De hecho, primero realiza la exponenciación, después la multiplicación y la división y por último, la suma y resta y efectúa todos los cálculos de izquierda a derecha. Luego en PRINT 3+4/2, primero se divide 4/2 y luego se suma tres.

El problema que se planteó al ordenador fue sumar dos números y luego dividir el resultado entre 2. Ninguno de estos ejemplos hace esto. Pero puede modificarse el orden en el que el ordenador efectúa los cálculos usando paréntesis, de la forma expuesta en los dos últimos ejemplos en la pantalla. En este caso, se realiza primero la suma comprendida entre paréntesis y, a continuación, se divide el resultado entre 2.

Conozca las limitaciones

Los números que el Spectrum puede manejar están limitados en cuanto a precisión y tamaño. Es improbable que la limitación en el tamaño plantee problemas. Los números positivos pueden tomar cualquier valor entre 4×10^{-39} (cuatro dividido entre uno, seguido de 39 ceros) y 10^{38} (1 seguido de 38 ceros). El Spectrum almacena estos números con una precisión de 9 ó 10 dígitos. El ordenador recuerda los nueve primeros dígitos, el resto los pone a cero.

Si se trabaja con números muy grandes se puede tropezar con otra de las argucias que utiliza el ordenador: el Spectrum no los muestra de la misma forma en que se teclearon. Por ejemplo, PRINT 2000000000000 produce 2E12 en la pantalla (la E significa exponente). Esto no es más que una forma abreviada de exhibir 2×10^{12} ó 2 seguido de 12 ceros. Tecleando PRINT 10, PRINT 100, PRINT 1000, se comprueba cómo maneja el Spectrum números crecientes.

EL PRIMER PROGRAMA

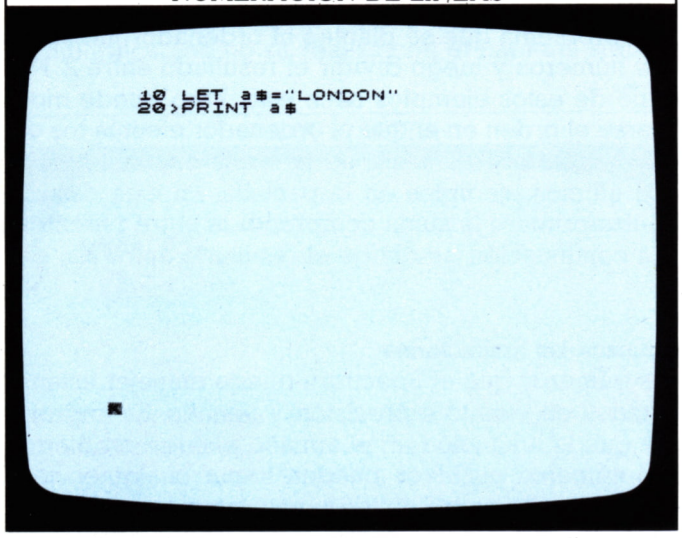
Hasta el momento, las órdenes que se han dado al Spectrum han provocado una respuesta inmediata. Estos mandatos han sido tan sencillos que, en muchos casos, hubiese sido más rápido prescindir del ordenador. Pero las órdenes por *sí* solas no son programas. El ordenador lee cada orden, la ejecuta y, a continuación, la olvida. Un programa es un conjunto ordenado de instrucciones que el ordenador almacena en la memoria. Puede ejecutarlas tantas veces como se desee y en el momento elegido.

De órdenes a líneas de programa

Una vez que se ha encontrado una tarea para el ordenador, hay que escribir el programa en pasos que el ordenador sea capaz de comprender. El Spectrum, como la mayoría de los ordenadores personales, utiliza un lenguaje de programación denominado BASIC (Beginners' All-purpose Symbolic Instruction Code). BASIC es un ejemplo de lenguaje de alto nivel, compuesto de palabras y símbolos familiares al usuario. Es por tanto un lenguaje fácil de aprender.

Las órdenes tecleadas pueden transformarse en líneas de programa si se numera cada línea escrita:

NUMERACION DE LINEAS

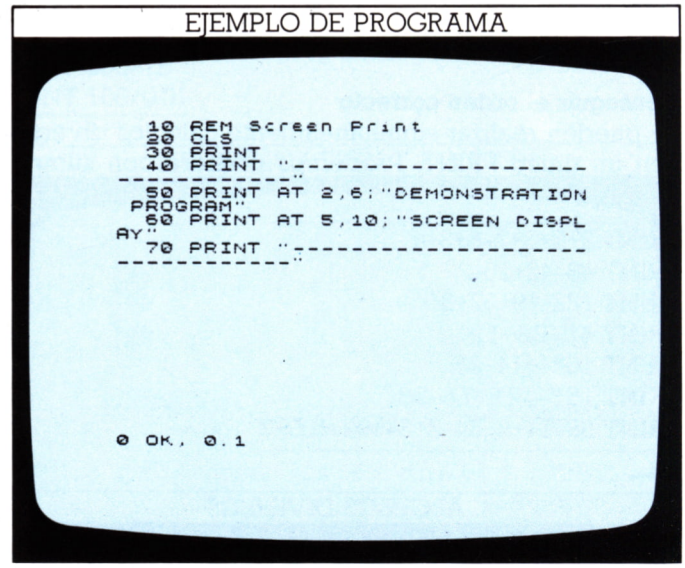


A medida que se teclea el programa, se observa que ahora no se ejecutan las instrucciones al accionar la tecla ENTER, sino que permanecen escritas en la pantalla.

Ahora que el programa está guardado en la memoria del ordenador, se puede ejecutar y ver qué hace. Esto se consigue accionando RUN y ENTER.

Puede resultar extraño que las líneas se numeren 10,20 y no 1,2. Frecuentemente, al escribir o probar un programa, se suele desear añadir o intercalar líneas. En el programa de la figura anterior, hay sitio para añadir líneas numeradas del 0 al 9 y del 11 al 19. El programa está todavía en la memoria del sistema. Pulse el botón RESET de la izquierda de forma que se reinicie el sistema. A continuación, compruebe lo que sucede cuando se ejecuta:

EJEMPLO DE PROGRAMA



Empezando por la primera línea, ¿qué es un REM? REM es la abreviatura de REMark (comentario). Es un instrumento muy útil a la hora de titular o etiquetar partes de los programas, de forma que se puedan encontrar rápidamente. Una vez que se programa con habilidad, se comprende la utilidad de las líneas REM a la hora de recordar cómo funciona un programa determinado. Las líneas REM permiten también que otras personas comprendan programas ajenos sin dificultad. Lo único que hace el ordenador con una sentencia REM es almacenar en su memoria.

Ya se ha visto que la palabra clave CLS es útil a la hora de borrar la pantalla. El uso de PRINT a veces (línea 30 de la figura) puede parecer absurdo. PRINT indica al ordenador que debe enviar todo lo que sigue a la pantalla y atender al principio de la línea siguiente.

Cómo corregir errores

Es fácil al teclear cometer errores que impidan que el programa funcione. El Spectrum no permite introducir una línea que contiene errores en BASIC. Sin embargo, no impide escribir programas cuyas líneas sean correctas pero que produzcan resultados erróneos. El ordenador siempre utiliza la versión más moderna de una línea; para modificar una línea entera basta pues con teclearla al final del programa y accionar la tecla ENTER; el ordenador insertará la nueva línea en el lugar adecuado.

Por qué es importante la puntuación

El programa siguiente utiliza las técnicas aclaradas en las páginas 16-17. Desconecte la fuente unos segundos y a continuación, teclee:

CALCULOS CON UN PROGRAMA

```
10 PRINT "3*18=";3*18
20 PRINT "6.25*4=";6.25*4
30>PRINT "179*9.8=";179*9.8

K
```

Para cada una de estas tres operaciones, la pantalla exhibe primero la operación que va a realizar y una vez que el ordenador efectúa los cálculos necesarios, el resultado. El punto y coma es muy importante. Garantiza que el resultado se mostrará en la misma línea (y justo a continuación) que la operación detallada. Si se prueba el mismo programa con una coma, o punto o nada en lugar del punto y coma, se comprobará la importancia de la puntuación a la hora de programar. El introducir la separación adecuada es también vital si se desea que el ordenador muestre cadenas y números legibles sobre la pantalla. El programa siguiente, que combina operaciones con la palabra clave PRINT, es un ejemplo de cómo aparecen los espacios introducidos en las cadenas cuando se ejecuta el programa:

PROGRAMA DE CONVERSION

```
 10 PRINT "CONVERSIONS"
 20 PRINT
 30 PRINT
 40 PRINT "1 foot=";12*2.54;" c
entimetres"
 50 PRINT
 60 PRINT "1 gallon=";8/1.76;"
litres"
 70 PRINT
 80 PRINT "10 kilometres=";10*5
/8;" miles"
 90 PRINT
100 PRINT "1 day=";24*60*60;" s
econds"

K
```

EJECUCION DEL PROGRAMA DE CONVERSION

```
CONVERSIONS

1 foot=30.48 centimetres
1 gallon=4.5454545 litres
10 kilometres=6.25 miles
1 day=86400 seconds


0 OK, 100:1
```

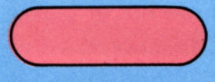
Cómo escribir un diagrama de flujo

Para que la ejecución de un programa sea correcta las operaciones deberán colocarse en el orden adecuado. Una forma útil de destacar los pasos involucrados en la tarea que se desea encomendar al ordenador es construir un diagrama de flujo. El diagrama siguiente muestra cómo debe estructurarse un programa que sume los mil primeros números. Cada bloque es una operación, y las flechas que conectan los bloques marcan el camino que debe seguir el programa. Tanto NUMBER como TOTAL representan variables numéricas-n y t. Este programa contiene dos nuevos conceptos: un «bucle» y un «punto de decisión».

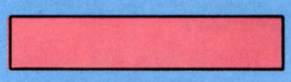
CONSTRUIR UN DIAGRAMA DE FLUJO

Muestra todos los pasos necesarios para la confección de un programa que sume los mil primeros números.

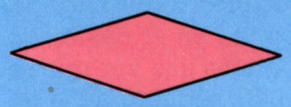
Clave



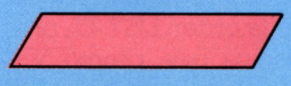
Terminador Comienzo o fin del programa



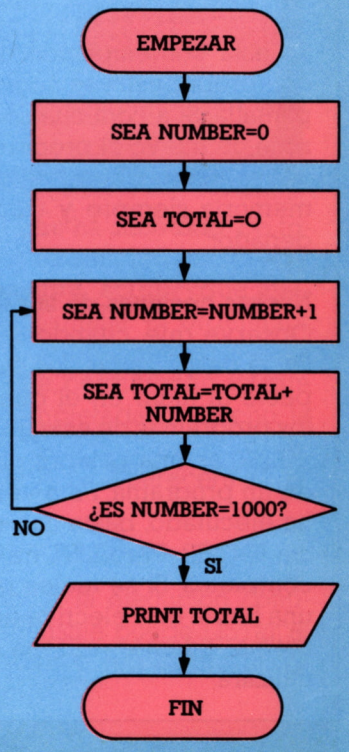
Instrucción Identifica cada operación por separado.



Decisión El ordenador debe hacer una decisión.



Entrada/salida El sistema debe tomar o enviar una información.



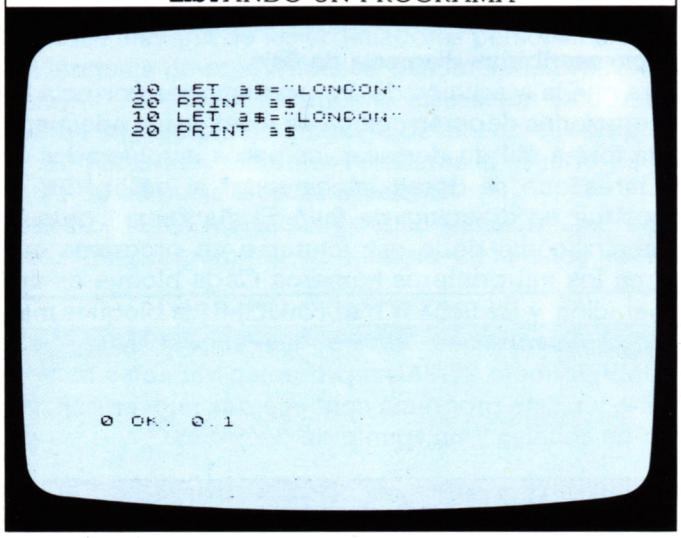
VISUALIZACION DE PROGRAMAS

Con frecuencia, al escribir programas se deseará hacer alguna comprobación o corrección. Con este fin, es deseable poder ver los programas en la pantalla una vez que han sido ejecutados. El Spectrum permite ver cualquier cosa almacenada en su memoria. En este caso, se desea ver el programa tal y como se tecleó.

Si Vd. enciende el ordenador de nuevo después de un breve descanso, teclee un programa de alguna página anterior.

La palabra clave LIST permite visualizar cualquier programa en la pantalla, que estuviera almacenado en memoria. Cada vez que pulse LIST el programa aparecerá en pantalla.

LISTANDO UN PROGRAMA



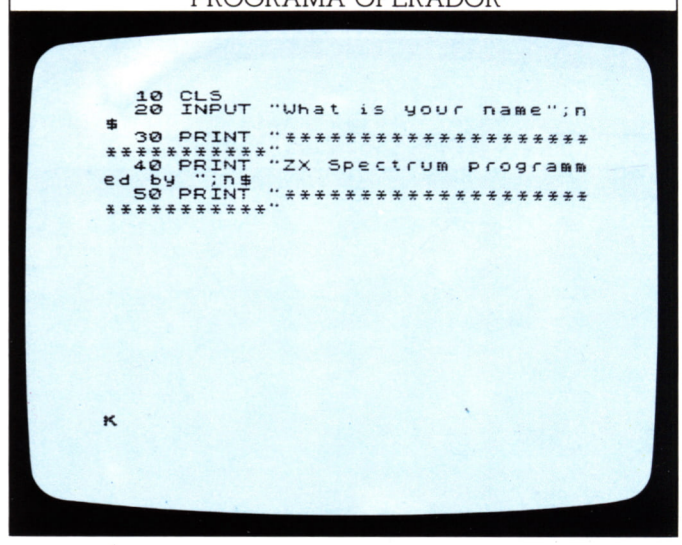
Se está contemplando una copia del programa. Para comprobar esto, se sugiere utilizar CLS para borrar la pantalla. Si a continuación acciona la tecla LIST, reaparece de nuevo el programa. Esto lo puede hacer tantas veces como quiera; el programa permanecerá en la memoria siempre y cuando no se desconecte el ordenador.

Nuevas opciones de la palabra clave LIST

Observe que, cuando se lista un programa, las líneas están desplazadas dos posiciones a la derecha con respecto al lugar en el que se teclearon. La mayoría de los programas de este libro aparecen en forma LISTada.

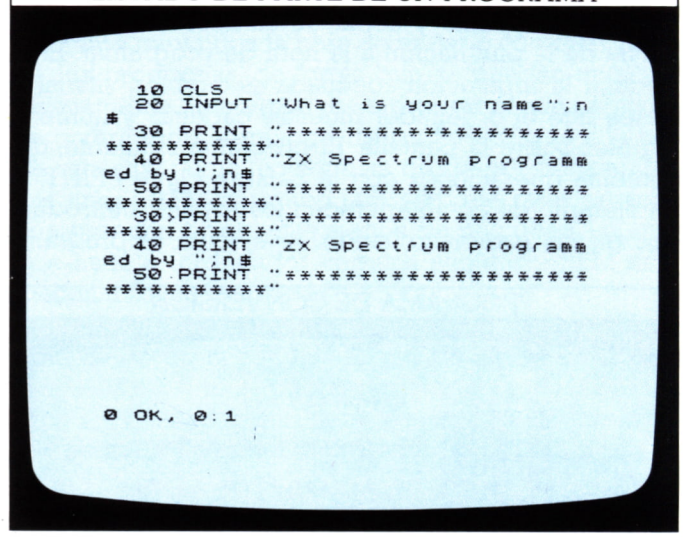
LIST es una palabra clave muy útil en el desarrollo de un programa. Para ver un programa una vez que se ha ejecutado, basta con teclear LIST seguido de la tecla ENTER. Pero LIST no se limita a exhibir programas completos. Si se teclea el siguiente programa se verá una forma más electiva de utilizar la orden LIST. EL programa, además, enseña una técnica que pronto Vd. utilizará.

PROGRAMA OPERADOR



Si ahora se desea visualizar el programa completo, accionar LIST. Pero puede que tan sólo quieran verse las últimas líneas de un programa largo. Si utilizando el programa del ejemplo, se teclea LIST 30, sólo aparecen las líneas a partir de la número 30:

LISTADO DE PARTE DE UN PROGRAMA



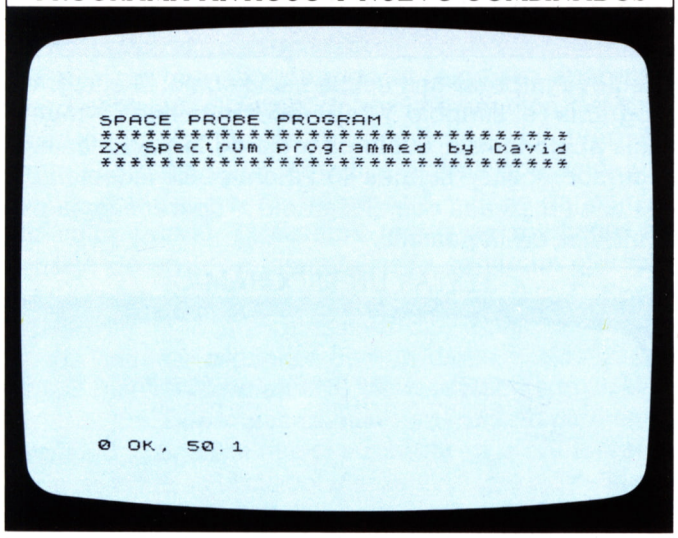
Cómo introducir un nuevo programa

Supongamos que se comienza un nuevo programa. Borre la pantalla y teclee la primera línea:

10 PRINT "SPACE PROBE PROGRAM"

y ejecútelo. Algo extraño sucede. El antiguo programa está todavía en memoria, de forma que el ordenador imprime la nueva línea 10 pero continúa ejecutando el programa anterior:

PROGRAMA ANTIGUO Y NUEVO COMBINADOS

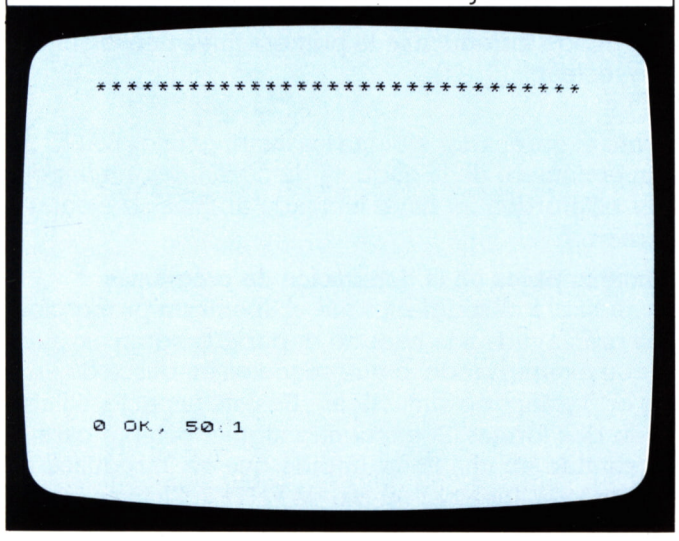


Hasta el momento, antes de introducir un programa nuevo Vd. ha pulsado el botón RESET. Evidentemente, hay formas mejores de deshacerse de los programas antiguos. Una posibilidad es utilizar la palabra clave NEW. Si se teclea NEW y después se acciona la tecla ENTER, el programa antiguo desaparece definitivamente.

Ejecutar (RUN) un segmento de un programa

Los programas pueden ejecutarse a partir de cualquiera de sus líneas. A menudo, encontrará problemas en la ejecución de alguna parte del programa. En un programa corto, es indiferente ejecutarlo entero o ejecutar sólo una parte. Pero, ¿qué sucede si la parte conflictiva está al final de un programa largo? El ejecutar el programa desde el principio supone un derroche innecesario de tiempo y esfuerzo. Afortunadamente, se puede saltar parte de un programa utilizando RUN y la etiqueta de una línea:

PROGRAMA PARCIALMENTE EJECUTADO



En esta figura, el programa operador se ha LISTado y a continuación se ha ejecutado desde la línea 50 (RUN 50). Esta orden hace que el ordenador se dirija a la línea 50, e imprima la línea final de asteriscos. Este uso de RUN limpia todas las variables de la memoria; el segmento que se va a ejecutar debe contener pues las líneas de definición de sus variables. Para comprender este efecto, teclee RUN 30 con el programa operador.

Mediante la palabra clave GOTO se consigue exactamente el mismo efecto. GOTO es una de las órdenes más simples y útiles del lenguaje BASIC. Cuando se utiliza sin el número de una línea de programa delante, GOTO provoca que el ordenador se dirija a la línea especificada y que, a continuación, se ejecute desde ese punto. Pero cuando GOTO forma parte de un programa, se obtienen resultados muy interesantes. Veamos un sencillo ejemplo:

```
10 PRINT "*"
20 GOTO 10
```

La pantalla debería configurarse de la siguiente forma:

USO DE **GOTO**

No hay que asombrarse por lo que ha sucedido. De todas formas, comprobará que no es sencillo borrar esta pantalla. La clave está en la palabra "scroll?" que aparece en la parte inferior de la pantalla. El ordenador pregunta si se desea que los caracteres impresos en la pantalla se desplacen hacia arriba, permitiendo así que continúe la ejecución aún no completada. De hecho, la ejecución de este programa no tiene fin. Para interrumpir el programa accionar la tecla N. Aparecerá el siguiente mensaje:

```
D BREAK — CONT repeats 10:1
```

Si se acciona cualquier otra tecla, la ejecución continuará. Cuando aparezca este mensaje puede LISTar el programa o bien borrarlo mediante la palabra clave NEW.

CORRECCION DE ERRORES

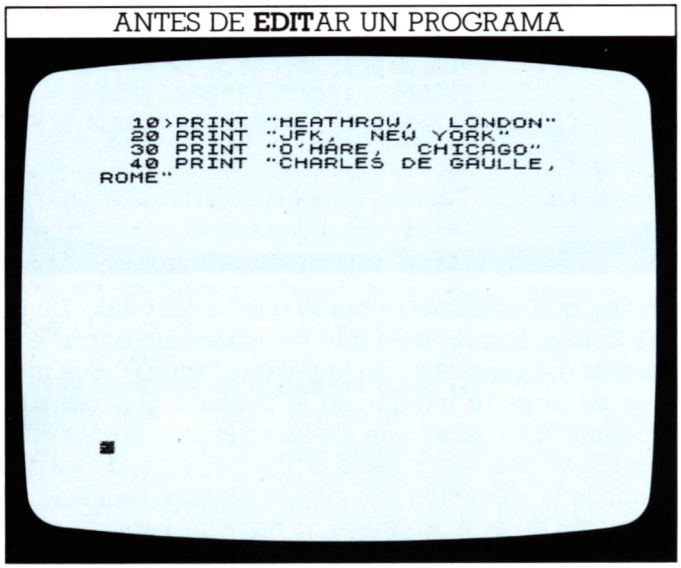
La programación es un pasatiempo en el cual los errores son inevitables. Es muy raro que un programa funcione satisfactoriamente la primera vez y, cuanto más largo, más difícil será que los resultados de la ejecución sean correctos. Es fundamental darse cuenta de que el hacer y corregir errores es una parte muy interesante del desarrollo de un programa. Los errores suponen una ayuda incalculable a la hora de aprender a hacer bien las cosas.

Por ejemplo, no puede modificarse la puntuación de un programa sin cambiar completamente el sentido de lo que se ha escrito. (Como ya se vio en la página 19, cualquier modificación en la puntuación provocará resultados drásticos.) Para el ordenador la puntuación significa algo muy preciso y si no es correcta, es posible que el programa no funcione.

Hay dos formas de modificar una línea de un programa. En primer lugar, se puede escribir encima; la nueva versión sustituye inmediatamente a la antigua en la memoria del ordenador. De todas formas, si se trata de pequeños errores, la reescritura supone una pérdida de tiempo. En este caso es conveniente utilizar los cursores para EDITar sobre la pantalla.

Editar en la pantalla

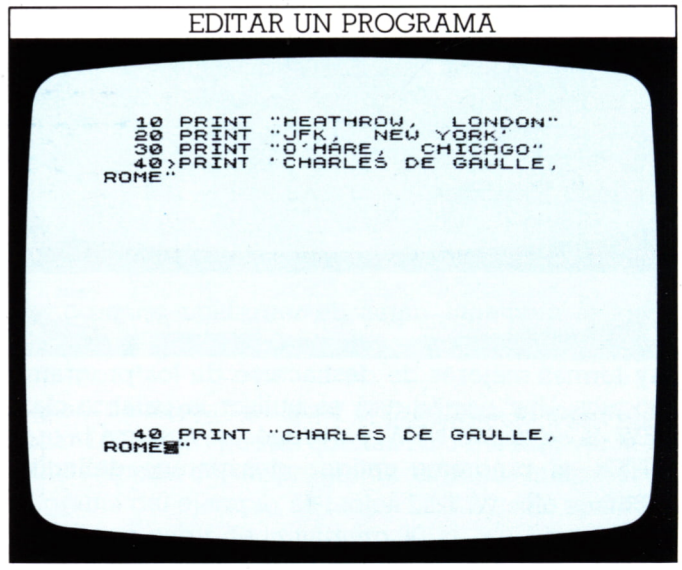
Para EDITar se utilizan las cuatro flechas de la parte baja del teclado y la palabra clave EDIT situada en la fila de enmedio. Veamos un ejemplo:



Se desea hacer la siguiente corrección (en la línea 40):

40 PRINT "CHARLES DE GAULLE, PARIS"

utilizando el editor de la pantalla. Primero teclee y ejecute el programa. Después borre la pantalla con CLS (V y ENTER) y LISTe el programa en pantalla pulsando K y ENTER. Cuando aparezca el listado, apriete la tecla que lleva impresa una flecha hacia abajo. El cursor del programa (el símbolo >) aparece en la línea y mueve hacia abajo la lista. Suelte la tecla de la flecha cuando el cursor alcance la línea 40. Ahora pulse la tecla EDIT y la línea marcada con el símbolo > aparece en la parte inferior de la pantalla.



A continuación pueden usarse las teclas de la primera fila del teclado, marcadas con flechas dirigidas hacia izquierda y derecha para mover el cursor a lo largo de la línea hasta el final de la sección que se va a suprimir, que está detrás de ROME, pero antes de las comillas. Ahora apriete DELETE hasta que desaparezca ROME. Sólo queda escribir los caracteres que van a reemplazar a la palabra ROME, y accionar la tecla ENTER. Aunque, de entrada, el proceso puede parecer complicado, pronto parecerá muy simple.

Con frecuencia deseará añadir líneas a un programa. Quizá no se tecleó CLS para empezar a introducir el programa en una pantalla limpia. Para el ejemplo anterior, puede introducirse la primera línea nueva sin más que teclear:

5 CLS

Como el ordenador ejecuta las instrucciones BASIC según el número de la etiqueta de cada línea, no importa que esta orden se haya tecleado al final, se ejecutará primero.

Primeros pasos en la depuración de programas

Ya se habrá descubierto que el Spectrum proporciona una gran ayuda a la hora de depurar programas, puesto que examina todo lo que se teclea en busca de errores ortográficos o sintácticos . En este aspecto, colabora de dos formas. Si encuentra alguna palabra carente de sentido en una línea, impide que se introduzca accionando la línea por medio de ENTER. El accionar ENTER no tiene ningún efecto.

En segundo lugar, ofrece una oportunidad de corregir el error, marcando la parte sospechosa de la línea con un signo de interrogación.

Es posible que aunque todo el texto sea aparentemente correcto, el resultado de la ejecución del programa no lo sea. Quizás se ha solicitado inadvertidamente al ordenador que realice alguna operación imposible, como sumar los números A y B, sin darles ningún valor previo. El sistema responde mediante un mensaje de error en la pantalla. Ya se vio un ejemplo en la página 14 —"2 Variable not found, 0:1". El Spectrum posee en total 25 mensajes diferentes.

Cada mensaje comienza con un dígito o letra (0-9 ó A-Z). El mensaje propiamente dicho está seguido de algo similar a 30:1, que indica que la ejecución se paró en la línea 30. El "1" significa que el error está en la primera sentencia de la línea. (Más adelante se verá que existe la posibilidad de escribir más de una instrucción en una sola línea.) A continuación se muestran algunos programas más avanzados que no funcionan. Ver la tabla de errores.

PROGRAMAS CON ERROR

```
10 FOR x=60 TO 100
20 PRINT 2↑x,3↑x
30 NEXT x

0 OK, 0:1
```

```
10 FOR f=1 TO 200 STEP 10
20 PLOT 150,f
30 DRAW 50,0
40 NEXT f

0 OK, 0:1
```

PROGRAMAS CON ERROR

```
10 INPUT "Enter number ";a
20 FOR f=1 TO 12
30 PRINT TAB 9;f;"*";a;"=";f*a
40 NEXT g

0 OK, 0:1
```

TABLA DE ERRORES

Estos son algunos de los mensajes de error que pueden encontrarse al escribir los primeros programas.

Código	Mensaje	Suceso
0	OK	No hay errores en el programa.
2	Variable not found	Se ha programado el ordenador para realizar alguna operación con una variable que no ha sido previamente definida. Esto puede ocurrir si se omiten las comillas antes y después de una cadena.
4	Out of Memory	Se ha completado toda la memoria disponible. Es improbable que suceda con el ZX Spectrum + a no ser que se intenten combinar varios programas largos con números de línea consecutivos.
5	Out of screen	El programa ha intentado leer (INPUT) más líneas de las que caben en la pantalla o ha intentado imprimir (PRINT) debajo de la última línea (ver págs. 15, 24-25).
6	Number too big	Como resultado de una operación se obtiene un número mayor que 10^{38} (ver página 17).
A	Invalid Argument	El programa ha intentado calcular una función con un argumento no válido —por ejemplo SQR seguido de un signo menos (ver página 17).
B	Integer out of range	El programa ha producido un número fuera del rango admitido por el ordenador para una operación determinada. Esto ocurre frecuentemente con los gráficos (ver página 28).
D	BREAK-CONT repeats	Aparece cuando se acciona N,SPACE o STOP en respuesta a "scroll?" (ver página 21). Este mensaje aparece también si se acciona BREAK cuando el ordenador está efectuando alguna tarea que no sea ejecutar un programa —por ejemplo cargar una cinta (ver página 60).
G	No room for line	Se ha llenado la memoria del ordenador y no hay sitio para la línea que se acaba de introducir (ENTER).
L	BREAK into program	Aparece si se acciona la tecla BREAK durante la ejecución de un programa. Los números de línea y sentencia que siguen al mensaje son los de la última línea que se ejecutó. Accionando CONTINUE se puede continuar la ejecución.

CONVERSACIONES CON EL ORDENADOR

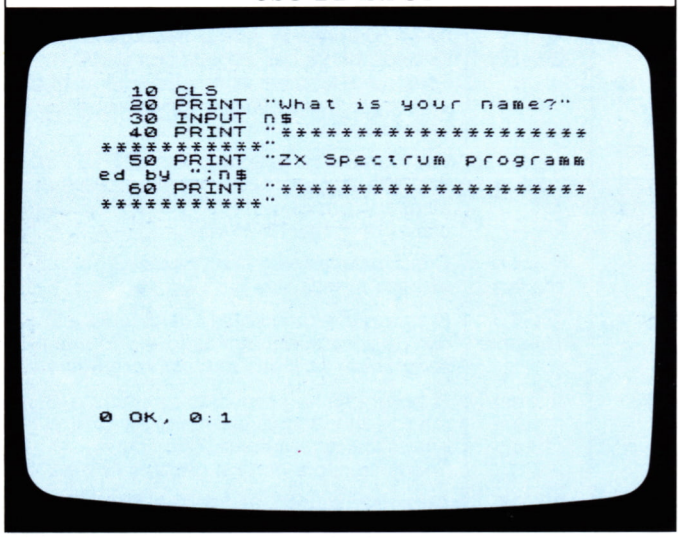
En todos los programas escritos hasta el momento, se ha dado al ordenador un conjunto de órdenes para ejecutar. Cada programa producía un único dato de salida, que era exactamente el mismo cada vez que se hacía una nueva ejecución. Pero la mayoría de los programas reales no son así; en los juegos, por ejemplo, los jugadores alimentan al ordenador con nuevas instrucciones. El sistema toma todas las instrucciones durante el juego y produce una información "salida" en respuesta a la "entrada".

Es difícil escribir un programa de cualquier complejidad que no se pueda interrumpir durante su ejecución para introducir nueva información.

El objeto de la palabra del BASIC INPUT es gestionar esta cuestión. Permite conversar con el ordenador —el usuario "habla" por medio del teclado y el ordenador "habla" por medio de la pantalla.

La palabra clave INPUT titula a la información tecleada —una variable numérica si es número, o una variable cadena si es una cadena— y hace que el ordenador la recuerde. La información se utilizará más adelante en un programa:

USO DE **INPUT**

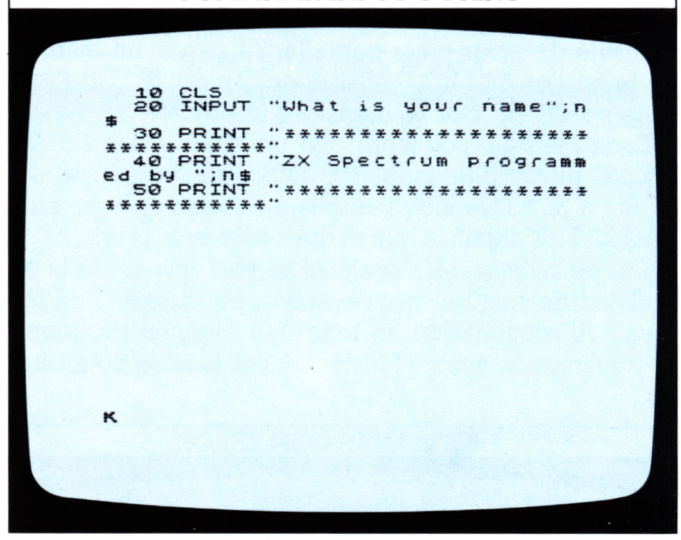


Preguntas desde el ordenador

El programa enseña al ordenador a exhibir la pregunta "¿Cómo te llamas?". La línea 30 interrumpe entonces la ejecución, dejando la pregunta impresa sobre la pantalla. A partir de este momento, el ordenador espera que se le dé información. No hay prisa; no hay límite de tiempo. El sistema esperará siempre o hasta que se le suministre la información deseada. Si se teclea un nombre y ENTER, la ejecución del programa continúa.

La línea INPUT toma el nombre y lo titula con la variable de tipo cadena n$. El signo $ muestra que el ordenador ha sido programado para recibir una variable de cadena. Este programa es similar al que se utilizó en la página 20 como ejemplo de LIST. Se puede comprobar en este programa que INPUT también imprime:

COMBINAR INPUT Y PRINT

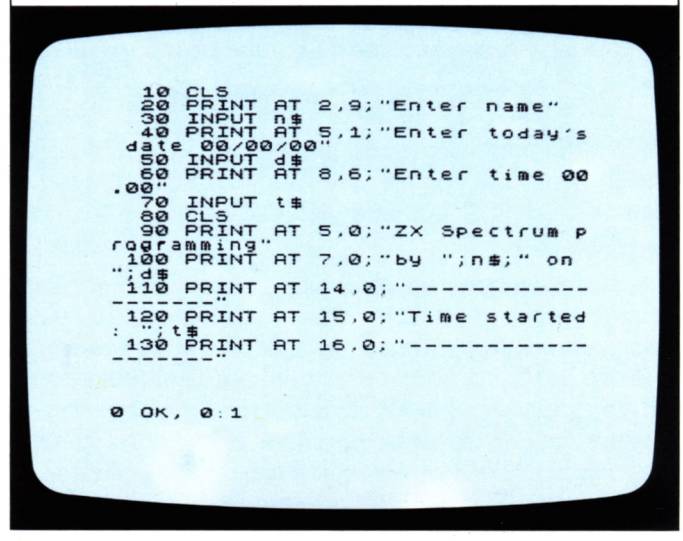


Programar con múltiples datos de entrada

Muchos programas utilizan INPUT varias veces para recoger distintos fragmentos de información. Para esto basta con elegir una variable diferente para cada sentencia INPUT. Una vez que se proporcione al sistema la información que cada variable titula, éste podrá usar todas las variables en un programa.

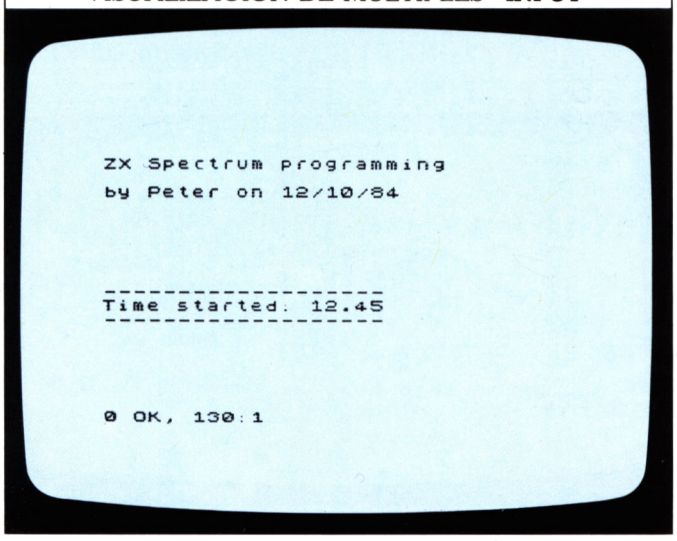
En el programa anterior, n$ titulaba una cadena de caracteres. Pero una cadena puede estar también formada por números. Si se titula un número como una cadena, el ordenador lo tratará como a cualquier variable de tipo cadena:

MULTIPLES DATOS DE ENTRADA **INPUT**



En la línea 100, el ordenador escribe d$, que es la fecha. Si la variable se hubiese llamado sólo d, el ordenador hubiese interpretado las líneas oblicuas como "dividir". Como cadena, d$ queda inalterado:

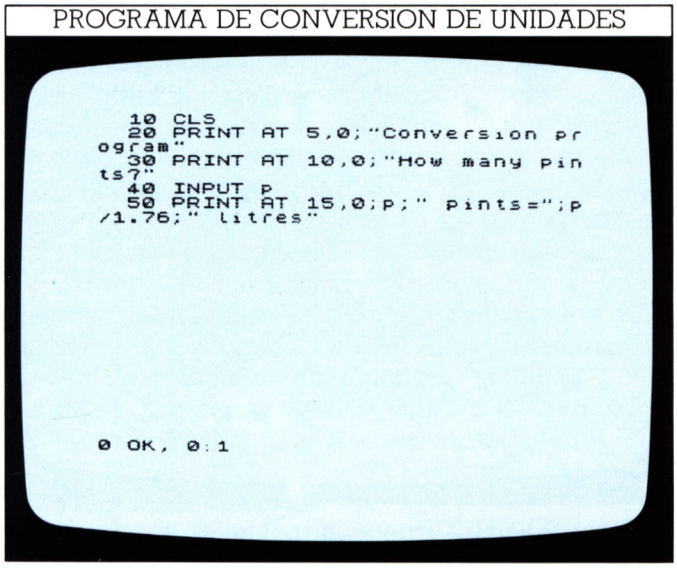
VISUALIZACION DE MULTIPLES **"INPUT"**



Uso de INPUT con números

Dado que INPUT admite también números durante la ejecución de un programa, tiene muchas aplicaciones prácticas. Considérese, por ejemplo, el problema de hacer una conversión de unidades de longitudes, tamaños o pesos. Los factores de conversión son siempre los mismos —2,54 cm una pulgada, 2,2 libras un kg; 1,76 pintas un litro, etc.— pero los números introducidos en cada nueva operación son diferentes. En la siguiente figura se muestra un sencillo programa:

PROGRAMA DE CONVERSION DE UNIDADES

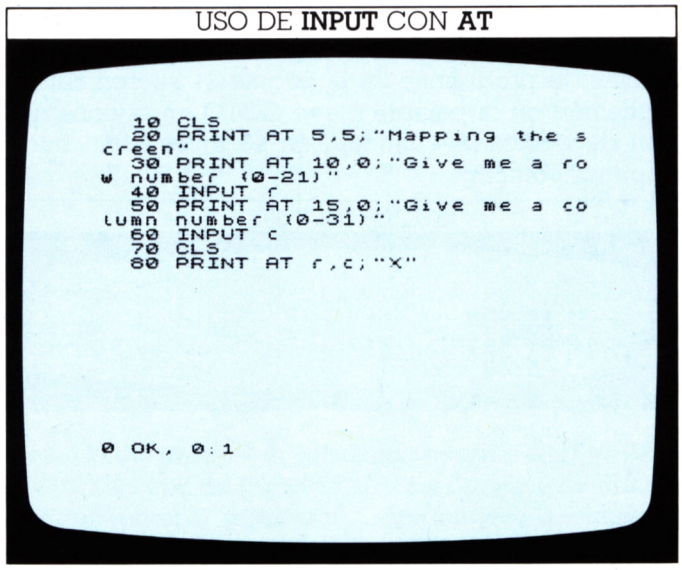


El programa pregunta cuántas pintas se desea transformar en litros, espera la respuesta, realiza los cálculos y exhibe el resultado sobre la pantalla. Dado que la línea INPUT espera un número como respuesta a su pregunta, la variable que produce, p, es de tipo numérico.

La letra p titula el número tecleado para cualquier uso posterior en un programa.

El siguiente programa solicita dos datos y utiliza la palabra clave AT para fijar un punto en la pantalla:

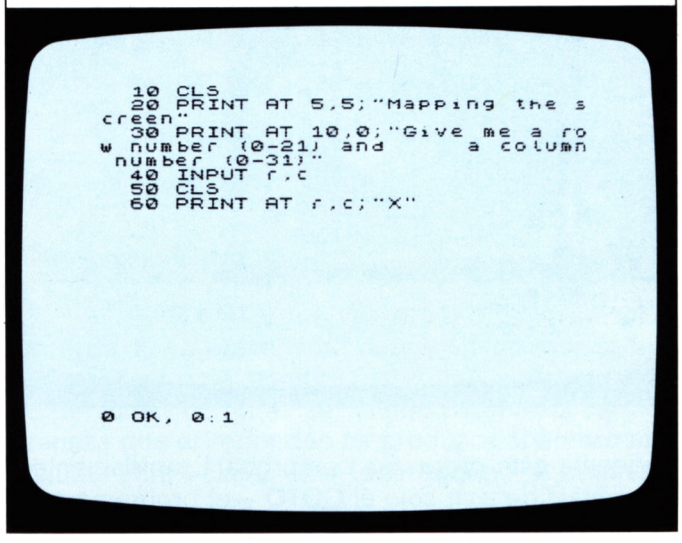
USO DE **INPUT** CON **AT**



Si se ejecuta el programa, se comprobará que pide una fila y una columna e imprimirá X en la posición indicada. Las letras r y c son variables numéricas a las que hay que asignar un valor. Estos valores son precisamente los que se teclean en tiempo de ejecución.

Se puede acortar el programa utilizando una sola sentencia INPUT. Escriba cada número seguido de ENTER:

LECTURA DE DOS VARIABLES CON UN **INPUT**

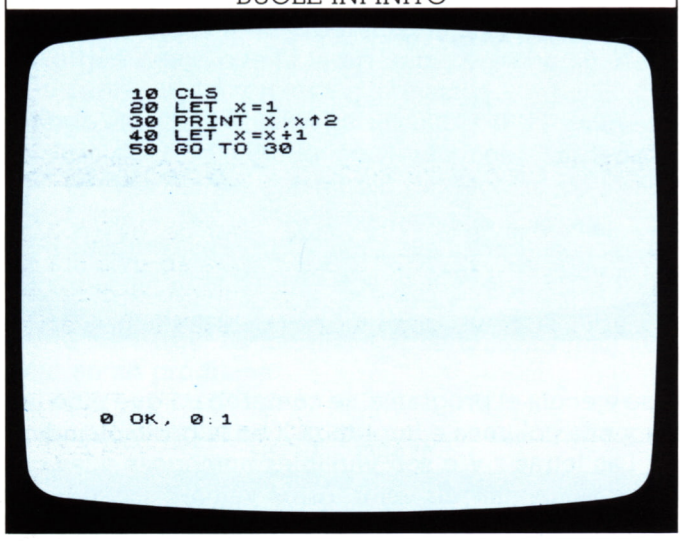


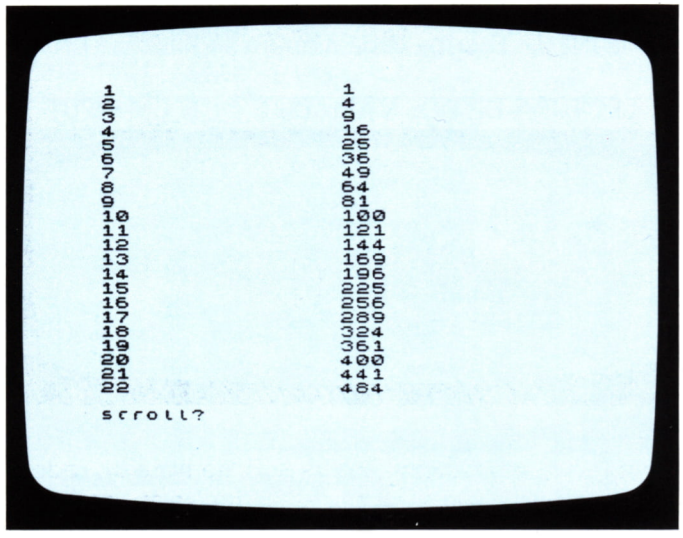
Los espacios entre "and" y "a column" pueden parecer extraños. Si no se introducen, se observará que el mensaje que imprime la línea 30 se reparte torpemente entre dos líneas. Los espacios sobrantes provocan que "a column number (0-31)" se escriba a partir de la mitad de la línea siguiente.

ESCRITURA DE BUCLES

Los ordenadores son muy eficaces a la hora de hacer tareas repetitivas simples. Para esto, necesitan algún medio que les permita ejecutar partes de un programa reiteradamente. Hay varias formas de escribir estos "bucles de programa". En la página 21 se comentó la aplicación de la palabra clave GOTO en la construcción de bucles. A continuación se muestra un bucle algo más complejo:

BUCLE INFINITO

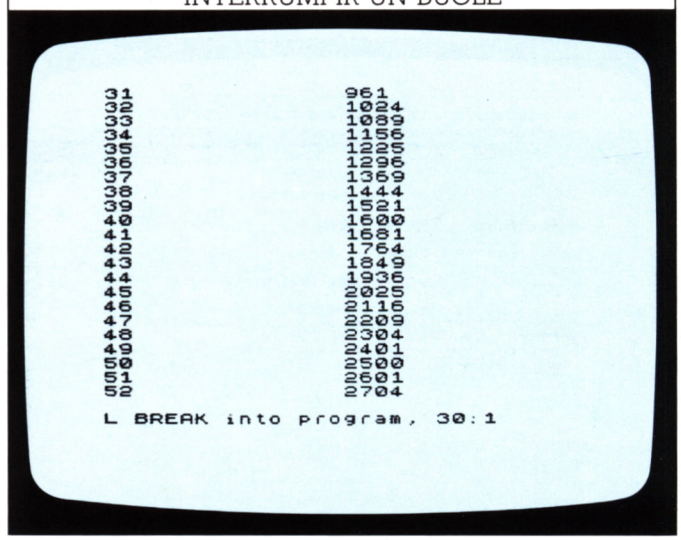




Si ejecuta este programa comprobará rápidamente la desventaja de usar sólo el GOTO —el programa es infinito. Cuando se completa una pantalla aparece "scroll?" en la parte inferior, indicando que aún no ha terminado la ejecución.

Si aprieta cualquier tecla, la ejecución continúa. Ya se vio en la página 21 que el bucle puede interrumpirse accionando la tecla N. BREAK también produce este efecto:

INTERRUMPIR UN BUCLE

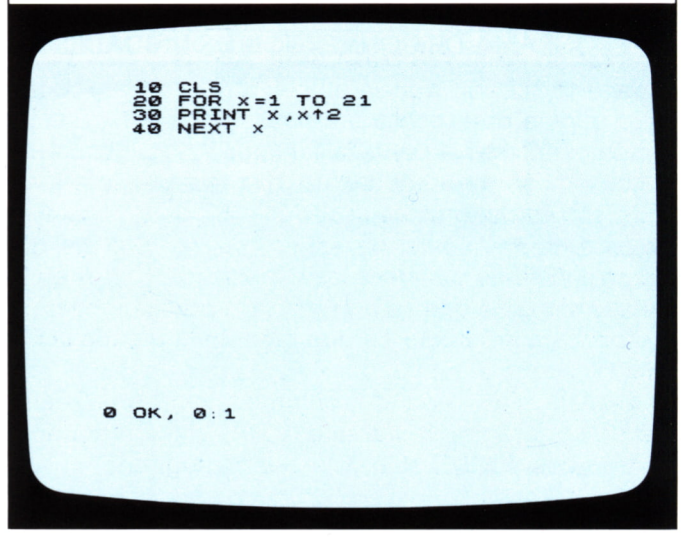


Vea que en este bucle, la línea 50 no hace volver al principio del programa, es decir a la línea 10. Si lo hiciera, x sería siempre igual que 1, y la pantalla se borraría cada vez que la primera línea fuera impresa.

Cómo parar un bucle

La solución a estos programas infinitos es el bucle FOR... NEXT. Permite limitar el número de veces que un bucle debe ejecutarse. Una adaptación del programa escrito con GOTO a FOR...NEXT:

BUCLE FOR...NEXT



El bucle FOR...NEXT no sólo mejora el programa sino que reduce una sentencia. Obsérvese que no hay que incluir LET X = sumar 1 a X en cada nueva ejecución del bucle; FOR...NEXT efectúa estas tareas automáticamente.

Comienza inicializando X a 1 e imprimiendo X y X cuadrado. La línea 40 pregunta por el siguiente valor

de x y así el programa comienza desde la línea 20 ejecutando las líneas siguientes otra vez. El proceso continúa hasta que X alcanza el valor 21, que es el límite impuesto en la línea 20.

Si fuese necesario, el bucle podría interrumpirse en cada paso, en espera de nueva información. Intente usar INPUT en un bucle FOR...NEXT:

INPUT EN UN BUCLE **FOR...NEXT**

```
  10 FOR n=1 TO 5
  20 CLS
  30 PRINT AT 5,5;"Temperature c
onversion"
  40 PRINT AT 12,0;"Type in a Fa
hrenheit temperature"
  50 INPUT t
  60 PRINT AT 14,0;t;" Fahrenhei
t=";(t-32)*5/9;" Centigrade"
  70 PAUSE 200
  80 NEXT n


0 OK, 0:1
```

Este programa transforma grados Fahrenheit en centígrados. El bucle FOR...NEXT comienza en la línea numerada con un 10 y limita el número de operaciones a realizar a cinco. La sentencia INPUT detiene en la línea 50 la ejecución, hasta que se introduce la temperatura Fahrenheit que se desea convertir. La línea 60 efectúa los cálculos y escribe los resultados.

Disminución de la velocidad de ejecución de un bucle

A menudo sucede que la ejecución de un programa es tan rápida que resulta imposible leer toda la información escrita en la pantalla. La línea 70 del programa de conversión de unidades de temperatura se encarga de este problema. La orden PAUSE detiene la ejecución temporalmente; de esta forma se dispone del tiempo suficiente para leer los resultados escritos en la pantalla antes de que se borre para comenzar una nueva ejecución del bucle. La duración de la parada la fija el número que acompaña a PAUSE; sus unidades son cincuentavos de segundo. Entonces PAUSE 200 interrumpe el programa durante 200/50s (4 segundos) mientras que PAUSE 10 indica 1/5 de segundo.

Cómo redondear números

La visualización de los resultados de la conversión puede mejorarse. No hay ningún problema si el resultado es un número entero, pero esto no es lo habitual. Cuantos más dígitos hay a la derecha del punto decimal más se desplaza la palabra "Centigrade" a lo largo de la línea, hasta llegar a partirse y pasar parte de ella a la siguiente línea:

CONVERSION DE TEMPERATURAS

```
Temperature conversion


Type in a Fahrenheit temperature
65 Fahrenheit=18.333333 Centigra
de
```

Para resolver este problema, escriba INT((t-32)*5/9+0.5) en vez de (t-32)*5/9. INT es la abreviatura de INTeger (entero) y transforma un número decimal en un entero. Si el resultado fuese 18.333333, la operación INT lo transformaría en 18:

VISUALIZACION DE LA CONVERSION REDONDEADA

```
Temperature conversion


Type in a Fahrenheit temperature
65 Fahrenheit=18 Centigrade
```

Recuerde que la función INT realiza siempre un redondeo por defecto del número real al que se aplica. El hecho de sumar 0.5 antes del último paréntesis cerrado garantiza que el redondeo se produzca al entero más próximo. Para aclarar esta idea, pruebe el siguiente ejemplo:

```
PRINT 3-1.1
PRINT INT(3-1.1)
```

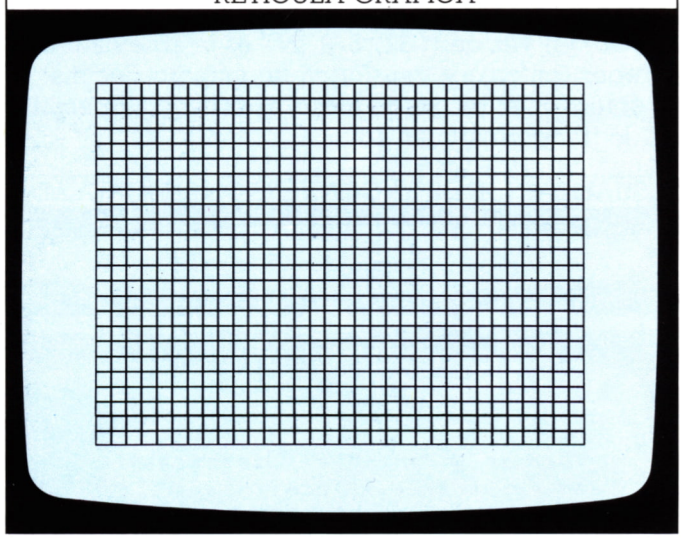
El resultado de la primera operación es 1.9 y el de la segunda 1. Pero 1.9 está de hecho más cerca de 2 que de 1. Al sumar 0.5 antes de aplicar INT se corrige este pequeño fallo.

EL TABLERO DE DIBUJO ELECTRONICO

El BASIC en el Spectrum incluye varias palabras clave para dibujar puntos y líneas en la pantalla. Para trazar un punto o una línea, hay que disponer de una forma adecuada de comunicar al ordenador dónde debe empezar a dibujar. La pantalla está dividida en una retícula, compuesta de pequeños puntos, denominados "pixels" *(picture elements)*.

Cada pixel tiene unas coordenadas asociadas —del 0 al 255, en horizontales y del 0 al 175 en verticales. La posición 0,0 está situada en la esquina inferior izquierda de la pantalla. Las coordenadas de un punto deben darse siempre en el orden x,y (x: horizontales, y: verticales). En la pantalla de la figura inferior, cada uno de los cuadrados elementales de la retícula es de 8 pixels de ancho por 8 de alto:

RETICULA GRAFICA



Si no se especifican las coordenadas deseadas, el Spectrum asume que se quiere comenzar a dibujar en el 0,0 (también denominado origen) o partir de la última posición visitada. Se puede generar un punto sobre la pantalla con la palabra clave PLOT. Para que aparezca un punto negro, teclee:

```
PLOT 128,88
```

Estas coordenadas puede Vd. comprobarlas en la página 61.

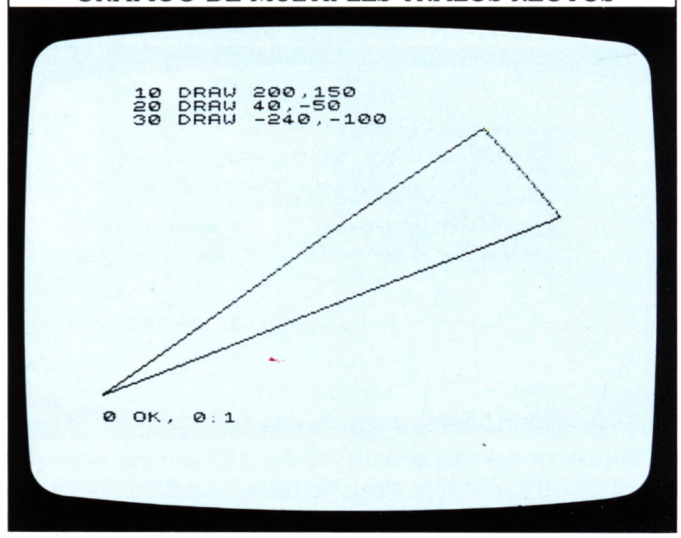
Cómo trazar líneas

Las líneas se trazan con la orden DRAW. Por ejemplo:

```
DRAW 128,88
```

Dibuja una línea desde el ángulo inferior izquierdo de la pantalla hasta un punto situado en el centro. Si a continuación se ordena al ordenador que trace una segunda línea, comenzará automáticamente a dibujar a partir del último punto visitado. Esta característica permite dibujar fácilmente figuras compuestas de trazos rectos:

GRAFICO DE MULTIPLES TRAZOS RECTOS

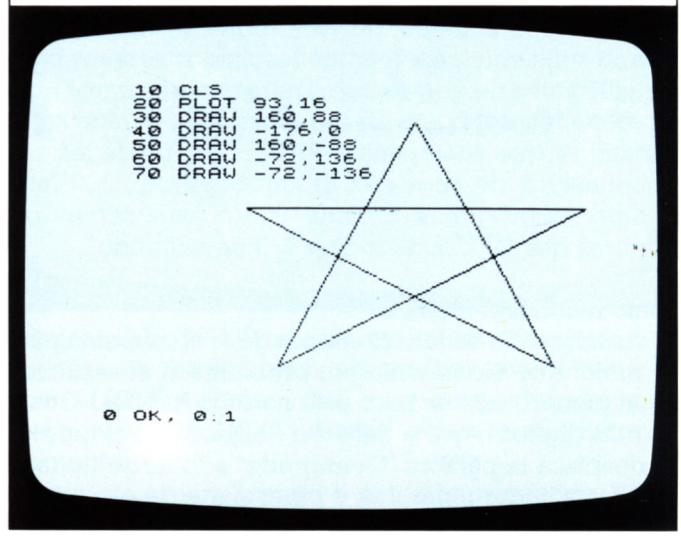


La línea 10 traza una línea desde el origen (0,0) hasta 200,150; el siguiente tramo está comprendido entre los puntos 200,150 y 240,100. Las coordenadas x e y varían en 40 y –50 respectivamente, de forma que la línea 20 del programa es DRAW 40,–50. Por último, la sentencia 30 traza el tercer lado del triángulo desde el punto 240,100 hasta el origen.

Eligiendo un punto inicial

Para dibujar una figura en medio de la pantalla hay que indicar al ordenador que se desea comenzar a pintar desde un punto distinto de 0,0. Primero ha de indicarse con PLOT dónde está el punto inicial de la figura. La línea siguiente se trazará desde este punto:

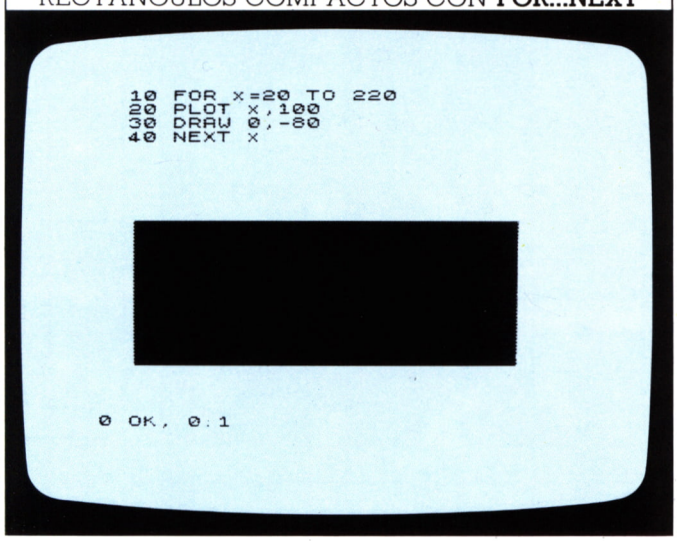
USO DE **DRAW** Y **PLOT**



Cómo rellenar figuras

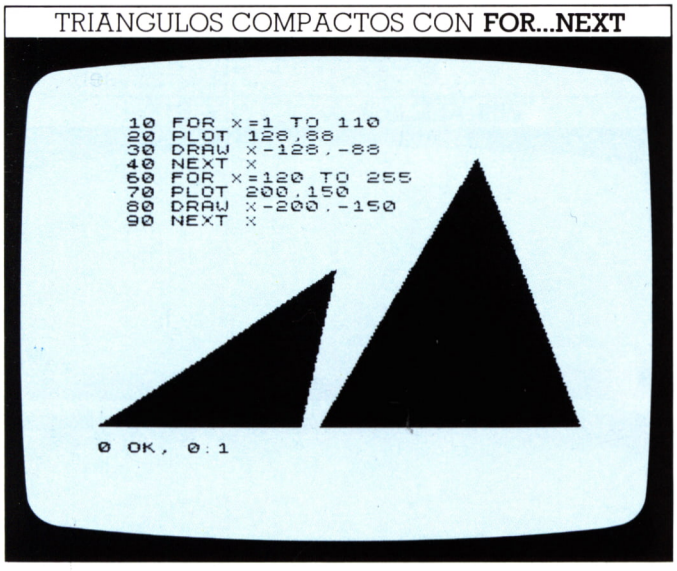
Ahora la cuestión es rellenar los dibujos de líneas para producir figuras compactas en negro:

RECTANGULOS COMPACTOS CON **FOR...NEXT**



Este programa traza líneas repetitivamente de arriba a abajo del rectángulo, avanzando gradualmente de izquierda a derecha. Un triángulo compacto se genera trazando líneas desde un único punto, que será uno de los vértices del triángulo:

TRIANGULOS COMPACTOS CON **FOR...NEXT**



Además de trazar puntos y líneas, el Spectrum puede generar caracteres gráficos que residen permanentemente en la memoria. Estos están impresos sobre las teclas 1 a 8 del teclado. El carácter grande es un cuadrado; los demás son mitades, cuartos y otras fracciones de un cuadrado.

Usando estos caracteres pueden generarse gráficos, pero las imágenes que se producen son poco finas y, dado que un carácter sólo puede ocupar posiciones propias de los caracteres, al introducir animación se producen movimientos discontinuos.

Para seleccionar caracteres gráficos, hay que introducir el cursor gráfico. Por ejemplo, para generar un cuadrado negro, teclee:

```
PRINT "■"
```

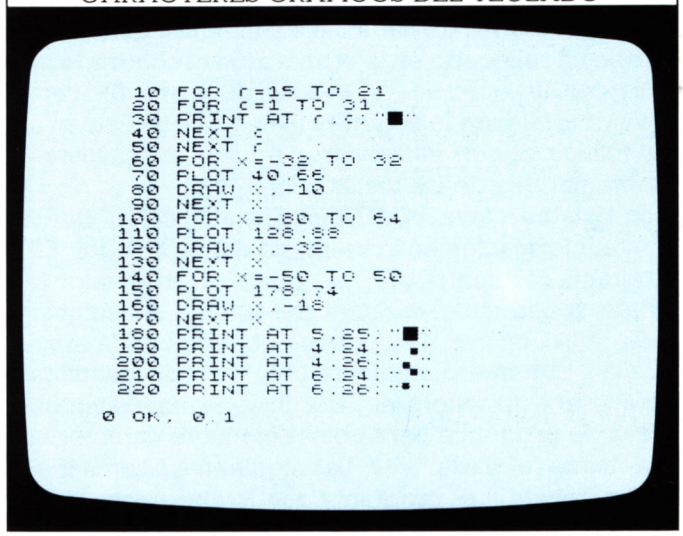
Para conseguir otra figura, antes de accionar la tecla 8 hay que pasar al modo gráfico (cursor G) pulsando GRAPH. Después, mantenga pulsado CAPS SHIFT, y apriete la tecla 8 para producir el cuadrado negro. Antes de continuar, pulse GRAPH de nuevo para desactivar el modo gráfico. Vd. puede conseguir el inverso (negativo) de cualquiera de los caracteres gráficos si no pulsa CAPS SHIFT de nuevo cuando presione la tecla del símbolo. Resumiendo, la secuencia de teclas necesaria para producir un cuadrado negro es:

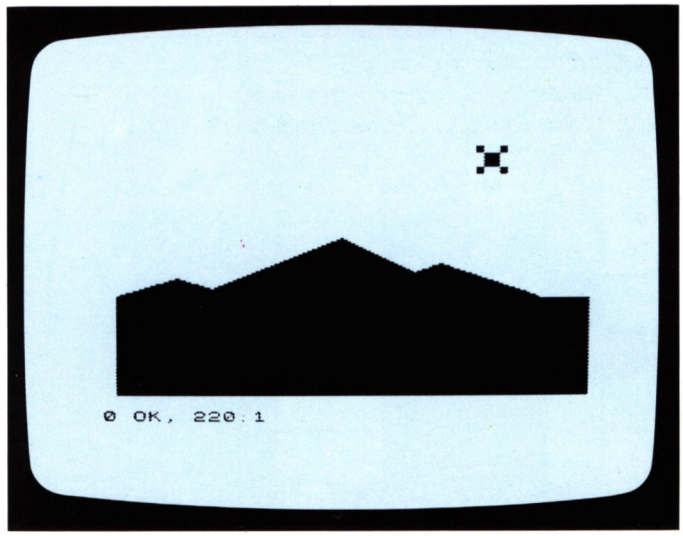
GRAPH
CAPS SHIFT Y 8
GRAPH

Dibujo de un paisaje sencillo

El siguiente programa es un ejemplo de cómo utilizar los símbolos gráficos, junto con las palabras clave PLOT y DRAW para dibujar un paisaje sencillo:

CARACTERES GRAFICOS DEL TECLADO





DISEÑO DE CARACTERES PROPIOS

Puede generarse cualquier forma gráfica sin más que utilizar PLOT y DRAW como se indica en las páginas 28 y 29. Resulta más eficiente almacenar caracteres propios en la memoria del ordenador, de la misma manera que el sistema almacena sus caracteres. Cualquier símbolo gráfico puede tratarse como un único carácter. Esto evita el tener que ejecutar un programa específico cada vez que se quiere generar un símbolo determinado. El Spectrum permite al usuario programar las teclas desde la A hasta la U con cualquier símbolo gráfico de su elección.

Cómo utilizar una retícula de caracteres

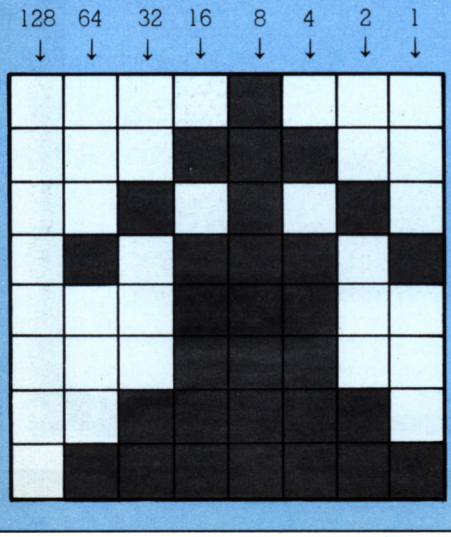
Supongamos que desea pintar una nave espacial. La forma elegida se visualizará sobre la pantalla como un conjunto de puntos de una red de 8×8; dibuje pues la figura deseada en un papel sobre una red de estas características o utilice la red de la página 61.

La figura debe dibujarse rellenando pequeños cuadrados en negro. Después sume los valores numéricos de los cuadrados de cada fila. En la figura del ejemplo, sólo se ha rellenado en la primera fila el cuadrado marcado con un ocho, luego el total para esta fila será 8; el valor total para la segunda fila es 28, etc. Estos valores totales deben introducirse en el sistema para reprogramar una de las teclas.

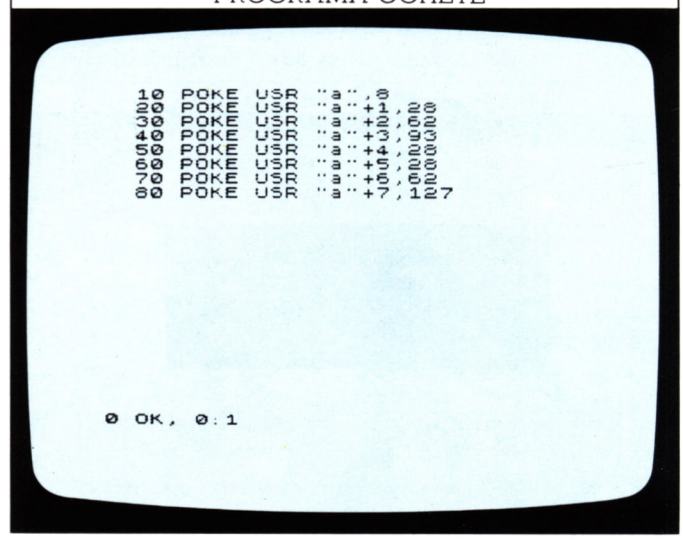
La palabra clave POKE permite almacenar directamente información en la memoria del ordenador. En el programa siguiente, USR "a" indica al ordenador que quiere generarse el carácter del usuario al accionar la tecla "a" (el mismo resultado se obtiene con la mayúscula A). El número siguiente (+1, +2, etc.) identifica la fila a la que se refiere el valor final. Se necesitan ocho líneas de programa para ubicar el nuevo carácter, que van desde "a" hasta "a"+7. Las siguientes figuras muestran el cohete y el programa que lo almacena.

CARACTER DE UNA SOLA RETICULA

Valor de cada cuadrado



PROGRAMA COHETE



Para generar el cohete, utilice la palabra clave PRINT. Para imprimirlo en medio de la pantalla, añada la siguiente línea de programa al programa de la figura anterior; previamente debe introducir el cursor gráfico y luego accione repetidamente la tecla A:

90 PRINT AT 15,8; "AAAAAAAAAAAAA"

VISUALIZACION DEL COHETE

Al conmutar el cursor gráfico, accionar repetidamente la tecla a y eliminar el cursor gráfico en la forma descrita en la página 29. Se ha introducido una línea de cohetes. Ya sólo queda ejecutar (RUN) el programa.

Cómo sumar caracteres

Lo primero que se observa es el reducido tamaño de los cohetes.

CARACTER MULTIRED

Total fila	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	Total fila
4						■			■	■			■				200
4						■			■	■			■				200
19				■		■	■	■	■	■			■	■		■	242
22				■		■	■		■	■			■	■		■	218
20				■		■			■	■			■				202
25				■	■		■		■	■		■		■		■	230
19				■			■	■	■	■		■			■	■	242
18				■			■		■	■		■			■		210
19				■			■	■	■	■		■			■	■	242
2							■		■	■		■					208
19				■			■	■	■	■		■			■	■	242
18				■			■		■	■		■			■		210
23				■		■	■	■	■	■		■		■	■	■	250
30				■	■	■	■		■	■		■	■	■	■		222
16				■												■	2
16				■												■	2

Si bien los caracteres definidos por el usuario están basados en redes o retículas de 8×8, un carácter puede ocupar más de una red.

De nuevo hay que teclear los valores totales asociados a cada fila en cada red de 8×8. Cada retícula se identifica mediante una letra. En el programa que sigue al cohete nuevo, las redes están marcadas con las letras a, s, d y f.

Diseño de pantallas completas con POKE USR

Es posible imprimir pantallas completas con caracteres diseñados por el usuario. En el programa siguiente, rellene los ceros de las líneas 10-80 para definir un carácter (la red de la página 61 supondrá una valiosa ayuda). En vez de asignar una posición fija al carácter, el programa imprimirá el carácter AT r,c donde r (número de fila) y c (número de columna), varían constantemente. A continuación se muestra el aspecto que ofrece la pantalla al ejecutar el programa, una vez que se han sustituido los ceros de las líneas 10-80 por los valores 195, 231, 126, 36, 36, 126, 231, 195:

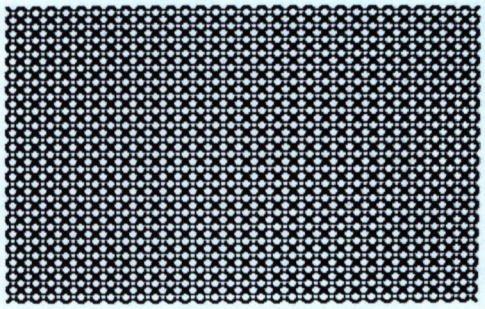
SUMA DE CARACTERES

```
 10 POKE USR "a",4
 20 POKE USR "a"+1,4
 30 POKE USR "a"+2,19
 40 POKE USR "a"+3,22
 50 POKE USR "a"+4,20
 60 POKE USR "a"+5,25
 70 POKE USR "a"+6,19
 80 POKE USR "a"+7,18
 90 PRINT AT 11,16;"A"
100 POKE USR "s",200
110 POKE USR "s"+1,200
120 POKE USR "s"+2,242
130 POKE USR "s"+3,218
140 POKE USR "s"+4,202
150 POKE USR "s"+5,230
160 POKE USR "s"+6,242
170 POKE USR "s"+7,210
180 PRINT AT 11,17;"S"
190 POKE USR "d",19
200 POKE USR "d"+1,2
210 POKE USR "d"+2,19
220 POKE USR "d"+3,18

scroll?
```

```
230 POKE USR "d"+4,23
240 POKE USR "d"+5,30
250 POKE USR "d"+6,16
260 POKE USR "d"+7,16
270 PRINT AT 12,16;"D"
280 POKE USR "f",242
290 POKE USR "f"+1,208
300 POKE USR "f"+2,242
310 POKE USR "f"+3,210
320 POKE USR "f"+4,250
330 POKE USR "f"+5,222
340 POKE USR "f"+6,2
350 POKE USR "f"+7,2
360 PRINT AT 12,17;"F"

0 OK, 0:1
```

PROGRAMA GENERADOR DE FORMAS

```
 10 POKE USR "a",0
 20 POKE USR "a"+1,0
 30 POKE USR "a"+2,0
 40 POKE USR "a"+3,0
 50 POKE USR "a"+4,0
 60 POKE USR "a"+5,0
 70 POKE USR "a"+6,0
 80 POKE USR "a"+7,0
 90 FOR r=1 TO 21
100 FOR c=1 TO 31
110 PRINT AT r,c;"A"
120 NEXT c
130 NEXT r

0 OK, 0:1
```



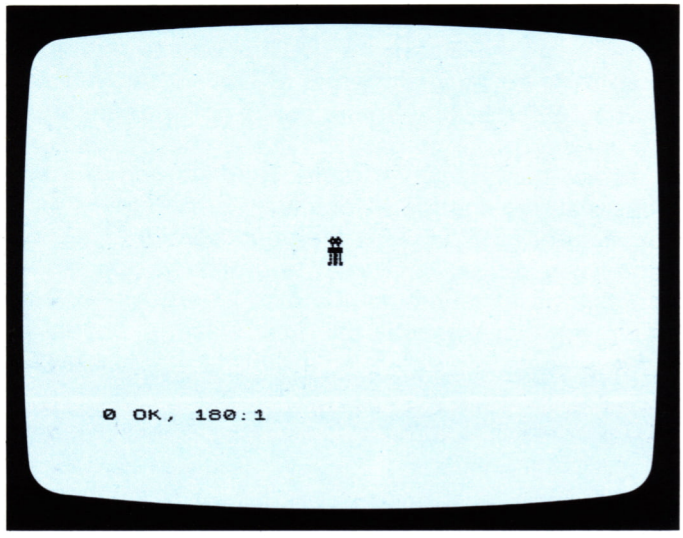
```
0 OK, 130:1
```

MOVIMIENTO

Una vez que ha aprendido a crear un carácter y a generarlo sobre la pantalla con las palabras clave PRINT y AT, puede intentar producir movimiento. En primer lugar, necesita disponer de un programa que genere un carácter:

CARACTER ESTATICO

```
 10 POKE USR "q",36
 20 POKE USR "q"+1,126
 30 POKE USR "q"+2,219
 40 POKE USR "q"+3,126
 50 POKE USR "q"+4,36
 60 POKE USR "q"+5,36
 70 POKE USR "q"+6,255
 80 POKE USR "q"+7,255
 90 POKE USR "r",255
100 POKE USR "r"+1,90
110 POKE USR "r"+2,90
120 POKE USR "r"+3,90
130 POKE USR "r"+4,90
140 POKE USR "r"+5,90
150 POKE USR "r"+6,219
160 POKE USR "r"+7,219
170 PRINT AT 11,15;"Q"
180 PRINT AT 12,15;"R"


0 OK, 0:1
```



Este programa crea un extraterrestre con dos símbolos gráficos, sin más que reprogramar las teclas Q y R. La línea 170 indica al ordenador que imprima la mitad superior del extraterrestre AT 11,15; mientras que la sentencia 180 usa AT para colocar la mitad inferior en el lugar adecuado.

Para mover el extraterrestre es un alivio saber que las líneas 10-160 inclusive permanecen sin alterar; no es necesario pues teclear NEW antes de introducir los siguientes cambios. Tampoco es necesario borrar las líneas 170 ó 180, ya que las nuevas sentencias las sustituirán.

Las siguientes líneas hacen que el extraterrestre se mueva (los símbolos gráficos reemplazarán automáticamente a los caracteres Q y R una vez que se ha ejecutado el programa anterior):

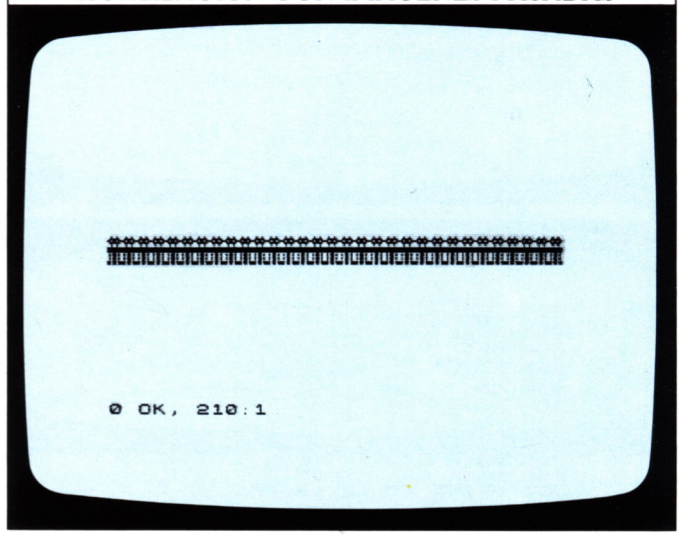
LINEAS DE MOVIMIENTO

```
170 LET r=11
180 FOR c=0 TO 30
190 PRINT AT r,c;"Q"
200 PRINT AT r+1,c;"R"
210 NEXT c


0 OK, 0:1
```

Cómo eliminar imágenes y controlar la velocidad

Cuando ejecute el programa modificado, observará que no hace exactamente lo que se deseaba. El Spectrum desplaza al extraterrestre de izquierda a derecha, pero no suprime las imágenes anteriores. El resultado obtenido es una larga línea de caracteres:

VISUALIZACION CON IMAGENES PASADAS



El problema es que no se ha comunicado al ordenador que debe eliminar las imágenes antiguas. Conseguirlo es muy sencillo, basta con imprimir un carácter en blanco después de cada carácter:

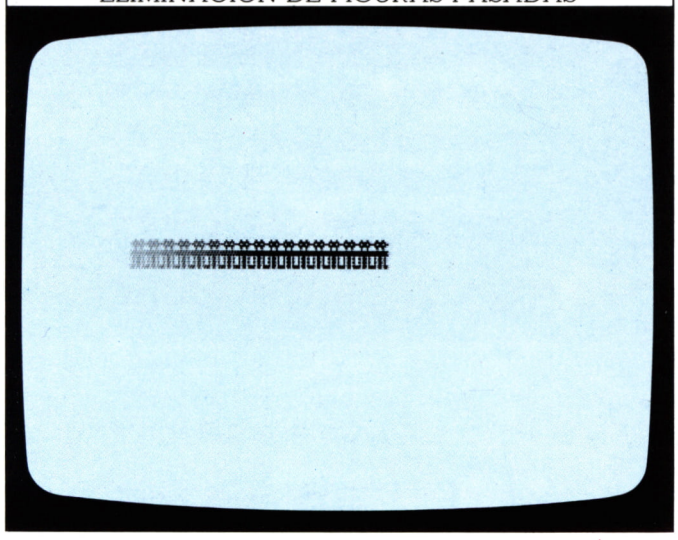
```
205 PRINT AT r,c-1;" "
206 PRINT AT r+1,c-1;" "
```

El extraterrestre se mueve de izquierda a derecha sin dejar rastro. Pero es muy rápido. ¿Cómo puede reducirse la velocidad? Con una retención en el tiempo.

```
207 PAUSE 2
```

Ahora el extraterrestre tarda más en alcanzar el borde derecho de la pantalla. La velocidad puede ajustarse alterando la sentencia PAUSE. La siguiente figura muestra una impresión de movimiento:

ELIMINACION DE FIGURAS PASADAS



Se apreciará que cuanto menor es la velocidad, mejor se visualiza el extraterrestre en la pantalla. Si el símbolo se mueve muy deprisa o si está compuesto de varios caracteres, se apreciarán vibraciones. Con un símbolo de 2 caracteres, hay una pequeña retención entre la impresión del primer carácter y la del segundo. Si se utilizan más de dos caracteres para construir un símbolo, se acentuará este efecto.

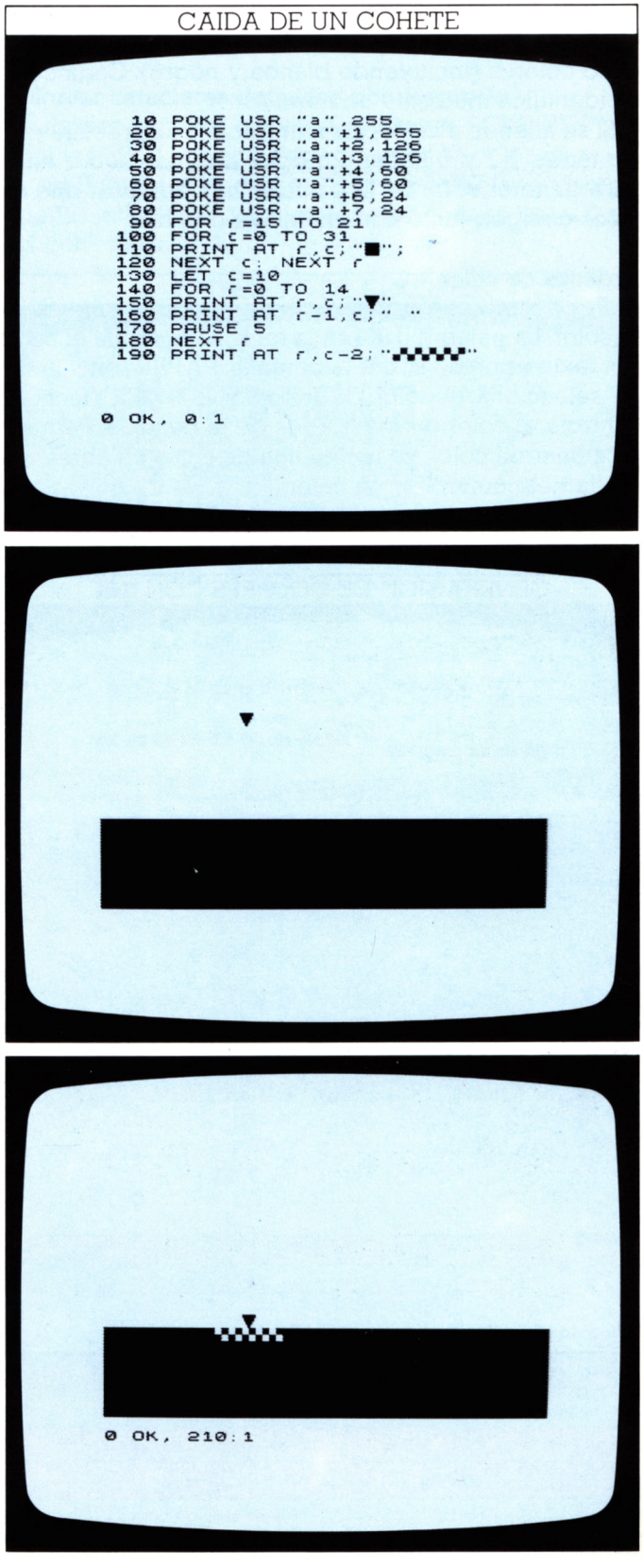
Movimiento de arriba a abajo

De igual forma, Vd. puede mover algo de arriba a abajo en la pantalla o viceversa. En vez de variar la posición horizontal (variable c en el programa del extraterrestre), se modifica la posición vertical con el mismo método.

El programa siguiente almacena un símbolo que representa un cohete en la tecla A y lo deja caer a lo largo de la pantalla. En la línea 140 se utiliza el mismo tipo de bucle FOR...NEXT que en el caso del extraterrestre (FOR...NEXT y otros bucles se explican en las páginas 26-27). La variable r (fila) aumenta de forma que el cohete se imprima progresivamente en posiciones más bajas.

Para asegurarse de que la nave no deje rastro, imprima un carácter blanco detrás. Dado que este programa utiliza un símbolo de sólo un carácter de anchura, basta con un único carácter blanco. Para que el cohete choque con algo, las líneas 90-120 dibujan un sencillo paisaje, similar al de la página 29. La línea 190, que sólo se ejecuta cuando se completa el bucle FOR...NEXT, imprime cinco caracteres gráficos (tecla 6). Con esta sentencia se produce el efecto del impacto de la nave.

CAIDA DE UN COHETE



EL COLOR

La visualización que produce el Spectrum en una pantalla de televisión no se limita a letras negras y símbolos sobre fondo blanco. Puede, de hecho, imprimir en ocho colores (incluyendo blanco y negro). Cada color se identifica mediante un número.

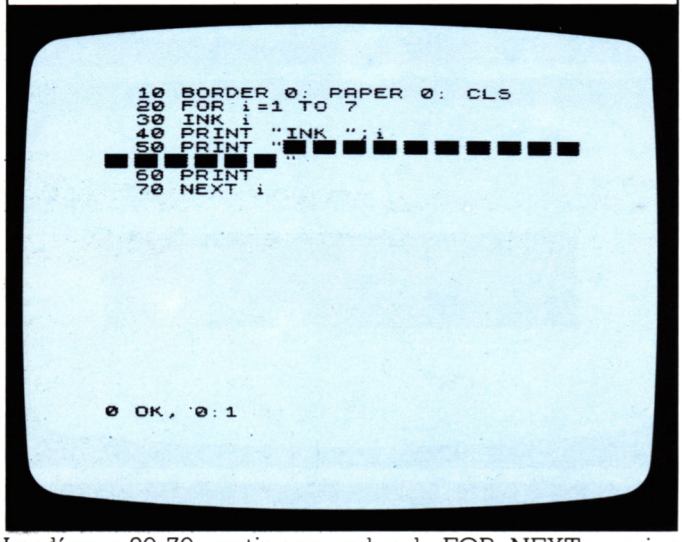
Si se atiende a las teclas numéricas, se apreciará que las teclas 1-7 y 0 tienen un color impreso sobre ellas. Para generar color en la pantalla, hay que usar uno de estos códigos, junto con una clave de color.

Ordenes de color

El Spectrum tiene tres formas diferentes de controlar el color. La palabra INK (en la tecla X) controla el color del texto impreso sobre la pantalla. PAPER (en la tecla C) selecciona el color de fondo, y BORDER (tecla B) controla el color de los bordes de la pantalla. Para seleccionar un color, se teclea una de estas palabras, seguida del número de un color.

Para ver los distintos colores que la orden INK genera, teclee el siguiente programa:

GENERACION DE COLORES CON **INK**



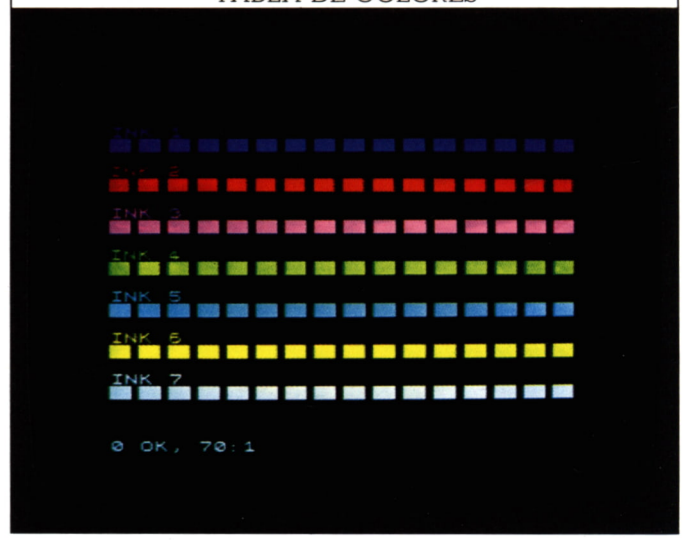
Las líneas 20-70 contienen un bucle FOR...NEXT, que imprime una fila de caracteres gráficos en cada uno de los colores codificados del 1 al 7. El programa exhibe todos los colores excepto el negro:

NUMEROS DE COLOR DEL SPECTRUM

Cada clave de color va seguida del número de color deseado

Número	Tecla	Color
0	Black	Negro
1	Blue	Azul
2	Red	Rojo
3	Magenta	Rosa
4	Green	Verde
5	Cyan	Celeste
6	Yellow	Amarillo
7	White	Blanco

TABLA DE COLORES

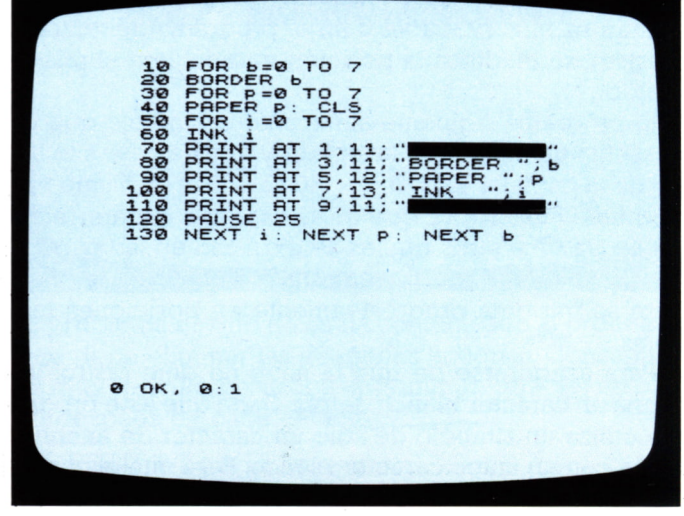


La sentencia 10 hace que el fondo y los bordes de la pantalla se coloreen de negro. Observará que contiene la palabra clave CLS. Esto garantiza un fondo negro antes de empezar a imprimir un texto. Normalmente, la palabra clave PAPER sólo produce un texto coloreado detrás de cada carácter impreso. Para colorear la pantalla completa, hay que introducir CLS después de PAPER.

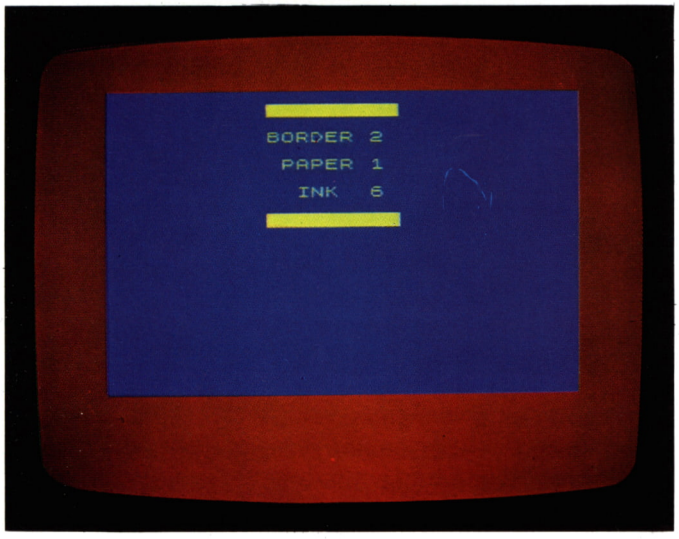
Cambio de tinta, fondo y borde

El paso siguiente consiste en ver qué sucede cuando se modifican los colores del texto, del fondo y de los bordes. El programa siguiente tiene esta misión; contiene tres bucles FOR...NEXT, cada uno de los cuales controla una de las órdenes de color. Al ejecutarlo, Vd. visualizará todas las combinaciones posibles de INK, PAPER y BORDER (¡Hay 512 en total!):

DEMOSTRACION DE **BORDER/PAPER/INK**



VISUALIZACION CON **BORDER/PAPER/INK**



Estas dos imágenes se producen al teclear el programa de demostración anterior. El listado de un programa puede escribirse con cualquier tinta, fondo y bordes, sin más que teclear una línea (sin etiqueta numérica) antes de introducir el texto. Negro sobre blanco se genera mediante la siguiente orden:

```
INK 7:PAPER 0:BORDER 0:CLS
```

Accionando NEW se reinicia el ordenador.

Mejorar la imagen

Si Vd. se siente decepcionado por la imagen obtenida, es posible que pueda mejorarla con unas sencillas comprobaciones. En primer lugar, compruebe que la televisión está correctamente sintonizada con la señal de salida del ordenador. La sintonización de ambos puede alterarse con el tiempo; por esto se aconseja realizar comprobaciones periódicas.

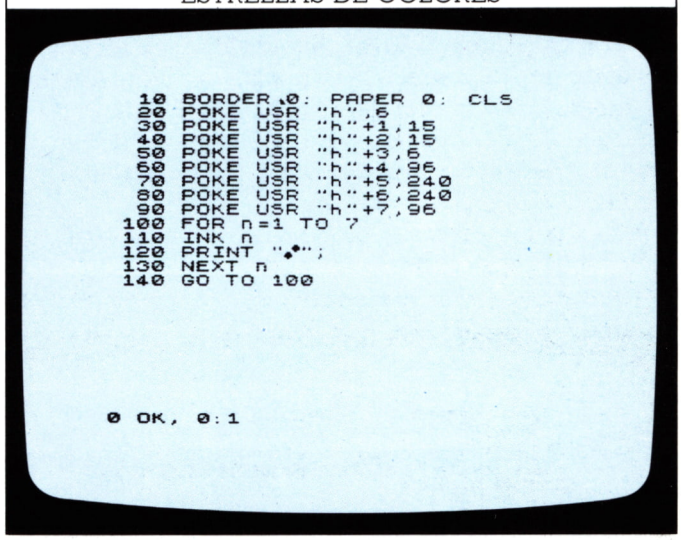
Si una vez que ha comprobado que la sintonización es correcta sigue obteniendo una imagen defectuosa, asegúrese de que no hay ningún otro dispositivo vecino que interfiera con la señal emitida por el ordenador —un video...

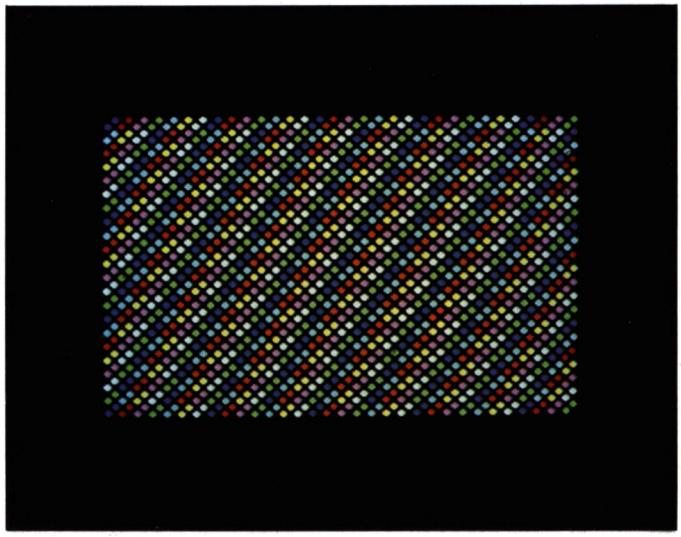
Finalmente, si una imagen es muy colorida y brillante, parecerá más distorsionada. Reduzca el color, la intensidad o el contraste para mejorar este aspecto.

Colorear caracteres definidos por el usuario

Cualquier carácter definido mediante POKE USR (ver páginas 30-31) puede imprimirse en el color deseado. El siguiente programa produce un carácter —dos pequeñas estrellas— y lo imprime en toda la pantalla en los colores del 1-7:

ESTRELLAS DE COLORES





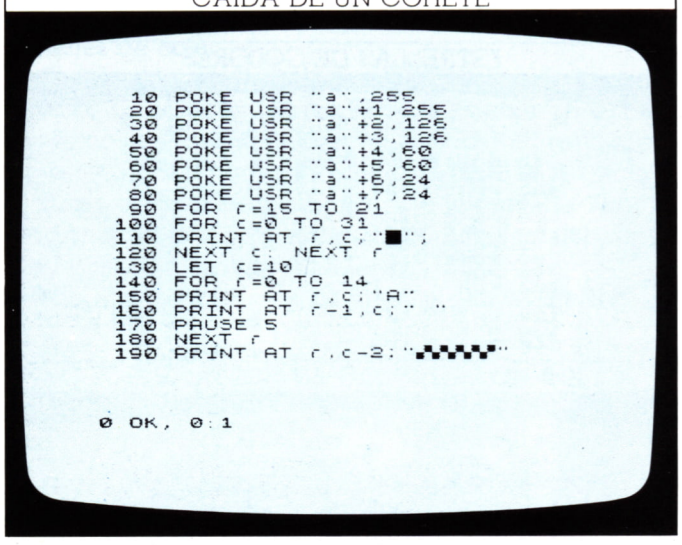
Una vez que Vd. se aficione al manejo de colores, puede obtener nuevas gamas por medio de efectos ópticos. Por ejemplo, cree caracteres propios a partir de una red de puntos en la que los puntos alternos tengan un color determinado.

A continuación, coloree el fondo en un color contrastante. Puntos rojos sobre fondo azul generarán un tono morado, el rojo y el amarillo producirán un naranja.

GRAFICOS EN COLOR

Los gráficos del Spectrum pueden colorearse con las mismas palabras clave y técnicas utilizadas para colorear textos y caracteres. No es necesario aprender palabras clave especiales ni escribir programas largos para generar gráficos en colores sencillos. Por ejemplo, el programa que provocaba la caída de una nave de la página 33. Si se teclea de nuevo el programa, puede colorear las imágenes generadas:

CAIDA DE UN COHETE



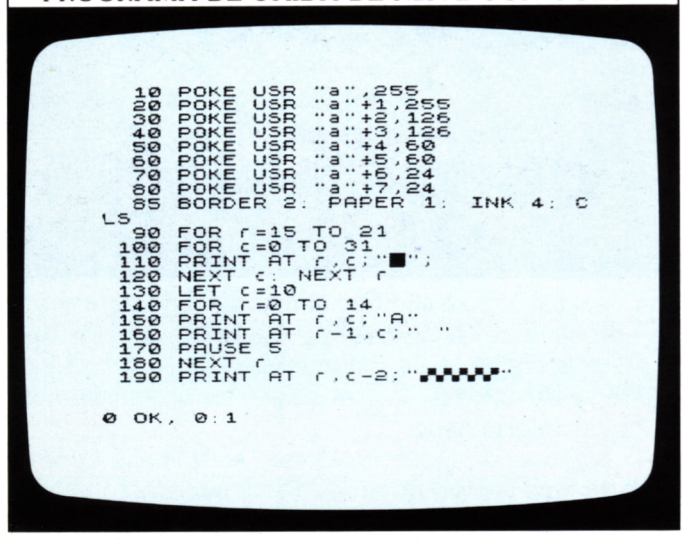
Programar gráficos en color

Añada la siguiente línea al programa:

```
85 BORDER 2:PAPER 1:INK 4:CLS
```

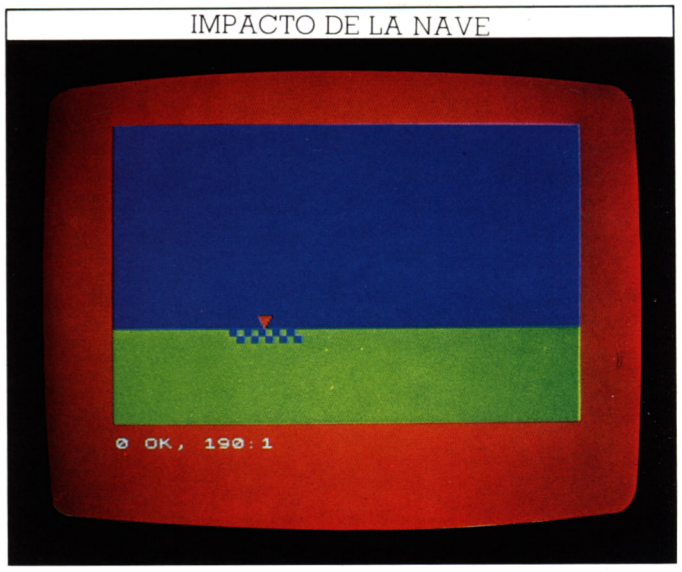
Si consulta los códigos de color de la página 34, averiguará el efecto que esta sentencia produce. Ejecute el programa. He aquí el programa:

PROGRAMA DE CAIDA DE NAVE CON COLOR



El programa, una vez modificado, genera la nave en rojo, el cielo en azul, el suelo en verde y un marco rojo. Cuando la nave llega al suelo, los caracteres gráficos que destacan el efecto del impacto se imprimen en azul sobre verde —esto es debido a que la línea 190 está también afectada por las palabras clave INK y PAPER de la sentencia 85. A continuación, se muestra la imagen obtenida una vez que el cohete ha chocado con el suelo:

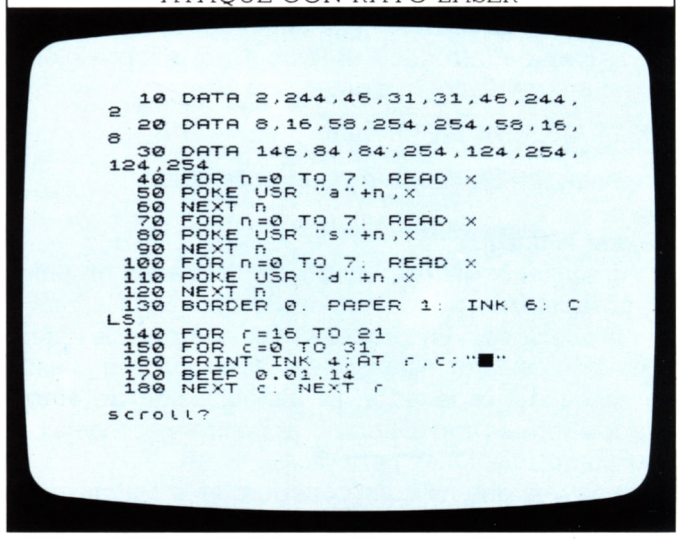
IMPACTO DE LA NAVE



Movimiento en color

Ya se tiene la experiencia necesaria para escribir un programa que genere movimiento en color. Como ejemplo, se incluye un sencillo programa que reune todos los elementos aprendidos hasta el momento. Está construido a partir de bloques separados o "módulos", fáciles de comprender:

ATAQUE CON RAYO LASER



ATAQUE CON RAYO LASER

```
 190 PRINT INK 2;AT 15,4;"▟"
 200 LET r=8
 210 FOR c=0 TO 20
 220 PRINT AT r,c;"▶▶"
 230 BEEP 0.08,-2: BEEP 0.02,12
 240 PRINT AT r,c;" "
 250 BEEP 0.02,8
 260 NEXT c
 270 PLOT 40,56: DRAW INK 7;128,
52
 280 BEEP 0.1,4
 290 DRAW INK 1;-128,-52
 300 FOR r=8 TO 15
 310 BEEP 0.02,320/r
 320 PRINT AT r,c;"▶▶"
 330 BEEP 0.08,420/r
 340 PRINT AT r,c;" "
 350 BEEP 0.02,320/r
 360 LET c=c+1
 370 NEXT r
 380 PRINT INK 7;AT 15,27;"▗▄▀▀▀
"

0 OK, 0:1
```

El lector se sentirá probablemente desconcertado por las líneas 10-30. Estas sentencias son, de hecho, una forma rápida de construir caracteres definidos por el usuario. Si se reprograman las teclas a, s y d por el método utilizado hasta el momento, se necesitarán 24 líneas de programa —8 líneas de órdenes POKE USR para cada letra. Con este nuevo método, cada tecla puede programarse con un máximo de cuatro letras; los valores totales de cada red se almacenan al ejecutarse cada sentencia DATA. El programa utiliza tres caracteres, dos para producir las partes delantera y trasera del cohete y una tercera para la base del láser.

La rutina READ... DATA, aclarada en las páginas 50-53, es un método rápido de reunir todos los datos necesarios en la construcción de gráficos definidos por el usuario. Permite poner todos los datos en una sentencia, en vez de usar múltiples líneas con POKE USR.

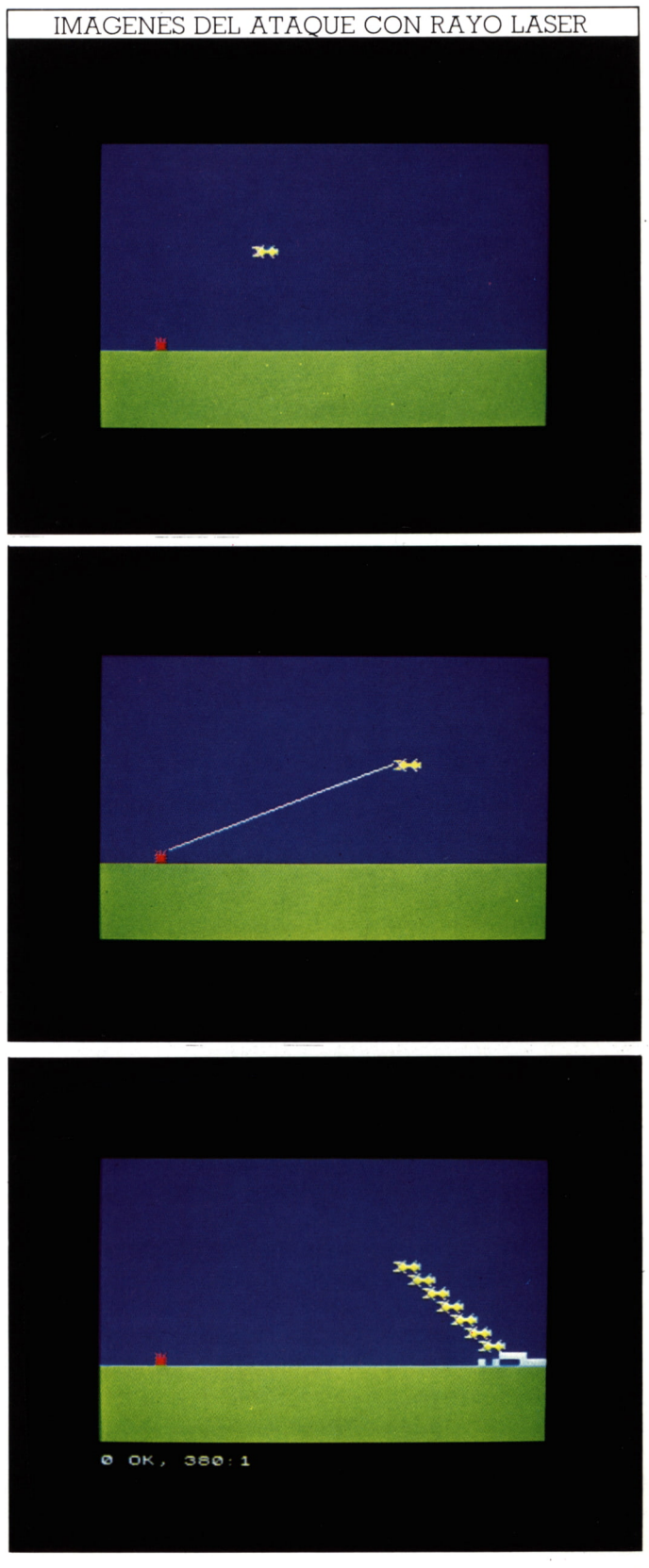
Las líneas 40 a 120 toman la información almacenada en las sentencias 10-30 y la transforman en los caracteres deseados. Las sentencias 140-180 generan el suelo de color verde, por medio de un doble bucle FOR... NEXT que imprime repetidamente un cuadrado verde a lo largo de la parte inferior de la pantalla. La sentencia 190 imprime la base del láser.

Las líneas 200 a 260 controlan el movimiento del cohete, enviándolo a lo largo de la pantalla a una altura fija. Se puede ver en la primera imagen de la figura de la columna de la derecha. Posiblemente no será necesario decir que la palabra clave BEEP produce el sonido del cohete.

Las líneas 270 a 300 programan el disparo del láser (imagen central de la figura). Una vez que se ha efectuado el disparo, las líneas 300 a 370 derriban el cohete, mientras que la última línea imprime su destrucción total. (En la última imagen de la derecha se muestra el cohete sin todas las retenciones que de hecho suceden).

Recuerde al teclear el programa que los caracteres gráficos se obtienen con GRAPH, y accionando la tecla apropiada definida por el usuario—a, s ó d. Suprima también el cursor gráfico antes de continuar.

IMAGENES DEL ATAQUE CON RAYO LASER



TECNICAS ESPECIALES 1

Ya vimos en las páginas 32-33 que es posible crear caracteres animados por el método de imprimir, borrar y reimprimir los caracteres en una nueva posición. Si se quiere dar un paso en las técnicas de generación de movimiento, pueden utilizarse dos palabras claves nuevas del Spectrum: INVERSE y OVER. Estas dos órdenes permiten producir movimiento cuando las técnicas habituales de impresión no funcionan.

La palabra clave INVERSE conmuta la estructura de puntos de la pantalla para producir el negativo del carácter. OVER permite imprimir un carácter sobre otro, de forma que se elude el borrado que normalmente ocurre al sobreimprimir.

INVERSE y OVER se utilizan para imprimir o dibujar algo sobre un fondo que normalmente tiene ya algo impreso o dibujado. Este programa muestra lo que sucede cuando se intenta producir acción sobre un fondo sin utilizar estas palabras clave. Dibuja un rayo láser que dispara a través de una red:

RAYO LASER Y RED

```
  10 BORDER 1: PAPER 7: INK 0: C
LS
  20 FOR y=16 TO 160 STEP 24
  30 PLOT 56,y: DRAW 144,0
  40 PLOT y+40,16: DRAW 0,144
  50 NEXT y
  60 PRINT INK 2;AT 20,1;"■"
  70 PRINT INK 2;AT 21,1;"■■"
  80 PLOT 16,16
  90 LET x=INT (RND*140): LET y=
INT (RND*160)
 100 DRAW INK 0;x,y
 110 DRAW INK 7;-x,-y
 120 GO TO 80


0 OK, 0:1
```

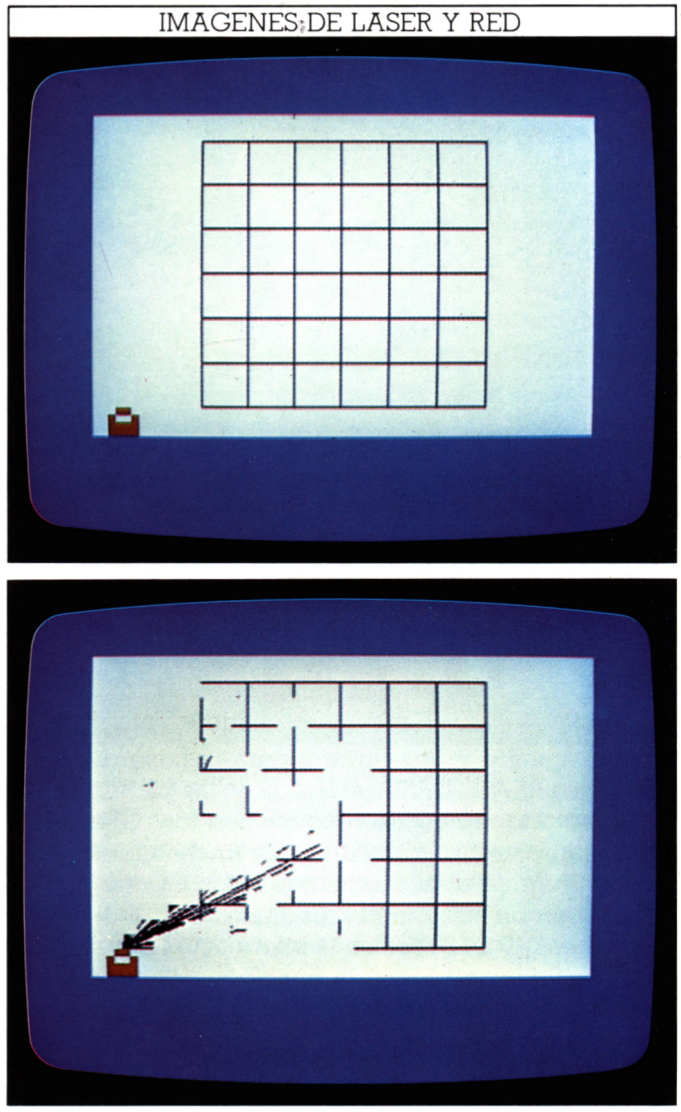
Las líneas 10 a 50 dibujan la red —red negra sobre fondo blanco con marco azul. Las líneas 60-70 imprimen un emisor de láser en la esquina inferior izquierda de la pantalla. La línea 80 marca un punto en la mitad de la parte superior del disparador de láser. Esta es una forma sencilla de mover el cursor gráfico hasta la parte superior del disparador, y situarlo en el punto del que saldrán los rayos.

La sentencia 90 produce los valores x e y que marcarán las coordenadas del punto al que se dispara —el punto final del rayo. Estas coordenadas se generan al azar por medio de la palabra clave RND (págs. 48-49).

La línea 100 traza un rayo negro hasta x, y, mientras que la sentencia 120 "desdibuja" el rayo dibujándolo en el color del fondo-blanco. La sentencia 80 devuelve el control al principio de la rutina de disparo.

Las siguientes imágenes muestran la imagen obtenida al principio de la ejecución e instantes después.

IMAGENES DE LASER Y RED



Desdibujar y sobreimprimir

En la segunda imagen se aprecia que el programa no hace lo que se esperaba. En primer lugar, cuando el rayo desaparece, se elimina también parte de la red situada en su trayectoria . En segundo lugar, y esto será quizá lo más sorprendente, cuando aparece un nuevo rayo, lo mismo sucede con los tramos de rayos anteriores que pasan por posiciones ocupadas por el rayo activo.

Los rayos antiguos pueden hacerse invisibles añadiendo INVERSE 1:

```
100 DRAW INVERSE 1; INK 0; x,y
110 DRAW INVERSE 1; INK 0; -x, -y
```

Pero ahora, no sólo se eliminan las imágenes de los rayos anteriores sino que también se suprime el rayo activo.

Es más, todavía aparecen cuadrados pequeños en la trayectoria del rayo que borra partes de la red. Pruebe:

```
100 DRAW INK 0;x,y
110 DRAW INVERSE 1; -x, -y
```

Ahora el efecto del disparo es correcto, si bien el punto final del rayo aún permanece impreso en la pantalla cuando se borra el resto. La red parece atravesada por líneas blancas situadas a lo largo de la trayectoria del rayo. La solución consiste en utilizar OVER:

```
100 DRAW OVER 1;INK 0;x,y
110 DRAW OVER 1; -x,-y
```

Ahora el disparo es totalmente correcto: el rayo aparece sobre el fondo de la red y desaparece sin dejar trazos blancos que marquen su trayectoria. De todas formas, el punto final del rayo sigue siendo visible. Si este punto coincidiese con una de las líneas de la red, dejaría una marca blanca. Esto sucede porque si bien el rayo pinta en una dirección y elimina en la dirección inversa a lo largo del mismo camino, el ordenador no sigue exactamente la misma vía en las dos direcciones. Para resolver este problema hay que recorrer el mismo camino en ambos sentidos:

```
110 PLOT 16,16: DRAW OVER 1;x,y
```

Se dibuja y elimina cada rayo desde el mismo punto origen (16,16) hasta el mismo punto final (x,y). Los puntos finales ya no permanecen sobre la pantalla. El programa funciona perfectamente. Los rayos atraviesan la red y desaparecen sin dejar ninguna evidencia de su existencia.

Sobreimprimir gráficos

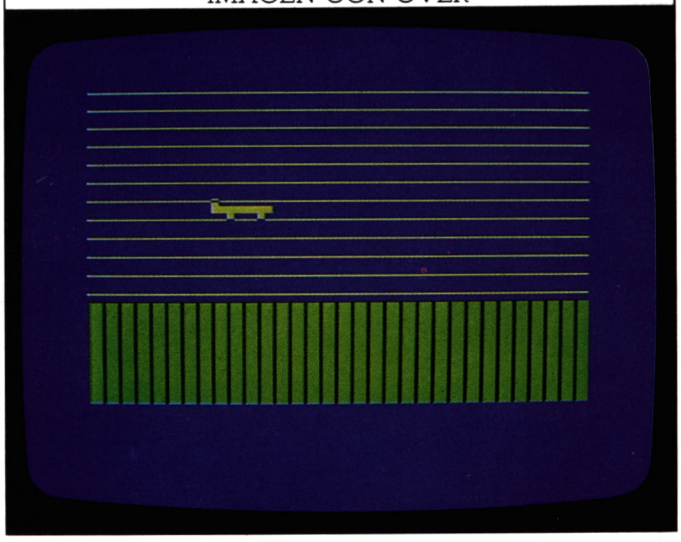
Puede sustituirse el rayo láser por un símbolo gráfico más grande, que se mueva a través de un fondo.

GRÁFICOS CON OVER

```
 10 BORDER 1: PAPER 1: CLS
 20 FOR r=15 TO 21
 30 FOR c=0 TO 31
 40 PRINT PAPER 4;AT r,c;" "
 50 NEXT c: NEXT r
 60 INK 0
 70 FOR x=0 TO 250 STEP 8
 80 PLOT x,55: DRAW 0,-55
 90 NEXT x
100 INK 6
110 FOR y=60 TO 170 STEP 10
120 PLOT 0,y: DRAW 255,0
130 NEXT y
140 LET r=13: LET c=3
150 PRINT OVER 1;AT r,c;"▬▬";
AT r+1,c+1;"▪ ▪"
160 PAUSE 5
170 PRINT OVER 1;AT r,c;"▬▬";
AT r+1,c+1;"▪ ▪"
180 LET r=r-1: LET c=c+1
190 IF r=0 THEN STOP
200 GO TO 150

0 OK, 0:1
```

Las líneas 20-50 imprimen un suelo verde sobre fondo azul. Las sentencias 60-90 trazan líneas en negro sobre el suelo verde para dar una mayor profundidad a la imagen. Las líneas 100-130 dibujan rectas amarillas sobre el cielo; la sentencia 110 controla el espacio que debe de haber entre ellas, a medida que se trazan de abajo hacia arriba sobre la pantalla. Las líneas 140 hasta el final del programa pintan un pequeño avión sobre el suelo y lo hacen despegar y volar (hacia la parte superior de la pantalla). Gracias a la palabra clave OVER, el fondo permanece inalterado. El símbolo en movimiento no tiene efecto sobre las líneas trazadas.

IMAGEN CON OVER



Para adquirir más experiencia en el uso de INVERSE y OVER, puede modificar los colores de los programas y sustituir las palabra clave PAPER por INK y viceversa. Al ejercitarse en el uso de estos mandatos en programas cortos se adquirirán nuevas ideas que más tarde podrán aplicarse a la confección de gráficos.

Recuerde que a las órdenes INVERSE u OVER los activa o desactiva el número que los acompaña.

Cómo realzar la intensidad del color de las imágenes

Además de generar imágenes en siete colores diferentes, el Spectrum permite obtener distintos matices de brillo en el color. La palabra clave que controla este parámetro es BRIGHT. Se utiliza en las sentencias de impresión (PRINT) de forma similar a INVERSE u OVER. Para obtener más brillo se utiliza BRIGHT 1 y para disminuirlo, BRIGHT 0. Si se teclea:

```
10 BORDER 0: PAPER 0: CLS
20 PRINT "without BRIGHT"
30 PRINT BRIGHT 1;"with BRIGHT"
```

Apreciará las diferencias que esta orden introduce. Puede usarse también BRIGHT 8. Esta palabra permite imprimir textos brillantes sin cambiar el color de la posición que se imprime.

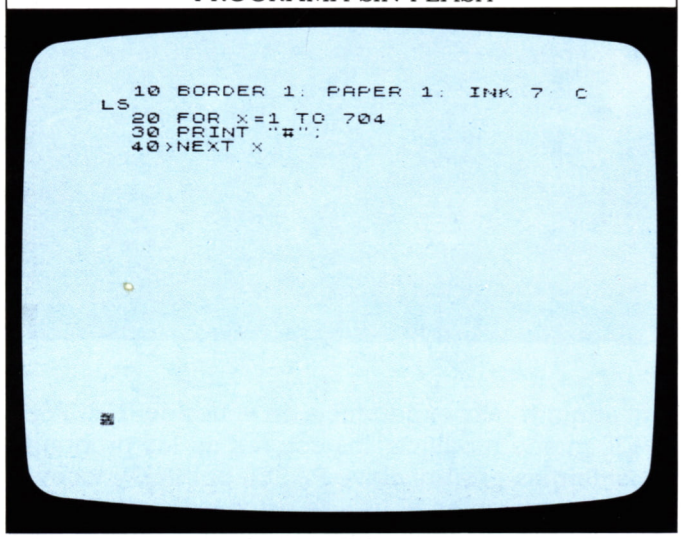
TECNICAS ESPECIALES 2

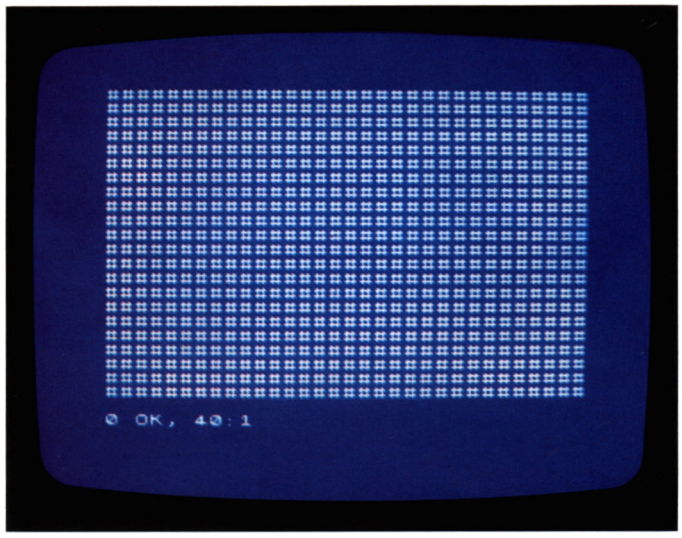
Parte del efecto de muchos programas radica en caracteres que aparecen y desaparecen fugazmente. Con el Spectrum puede conseguirse este efecto cambiando los colores del texto y del fondo rápidamente. Sin embargo, no conviene precipitarse; hay una forma mucho más eficiente y rápida de obtener caracteres fugaces. El Spectrum dispone de la palabra clave FLASH, que produce el efecto deseado.

Cómo obtener caracteres fugaces

FLASH puede tomar dos valores numéricos, 0 y 1. FLASH 1 genera un carácter fugaz, mientras que FLASH 0 detiene el proceso. El siguiente programa muestra el efecto que produce FLASH. Introdúzcase primero el programa sin sentencias FLASH:

PROGRAMA SIN FLASH





La línea 10 produce un marco y un fondo azules y texto en blanco. El rango de valores de la línea 20 (de 1 a 704) se elige porque existen 704 posiciones de caracteres en la pantalla de televisión (22 líneas de 32 caracteres). Si se añade la siguiente sentencia, la figura aparecerá fugazmente:

```
10 FLASH 1:BORDER 1:PAPER 1:INK 7:CLS.
```

Al ejecutar este programa, el texto se colorea alternativamente y la figura aparece y desaparece. Una pantalla llena de caracteres fugaces se observa difícilmente, y después de un rato Vd. querrá detener la ejecución. Si teclea:

```
FLASH 0
```

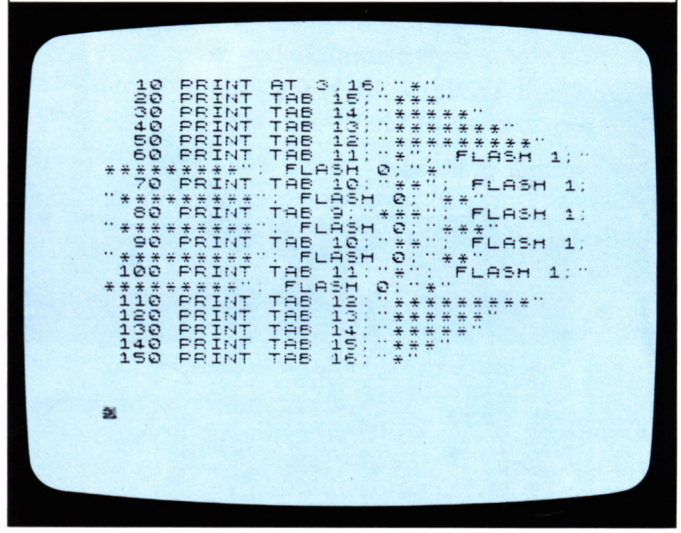
No sucede nada nuevo. Esto es debido a que la sentencia FLASH sólo afecta a los caracteres visualizados después. Si en vez de este, se teclea CLS, la figura cesa de aparecer y desaparecer.

FLASH puede utilizarse en una sentencia PRINT de la misma forma que OVER y BRIGHT. Todas estas palabras clave pueden usarse de dos formas. Si ahora se suprime FLASH 1 de la línea 10 y se teclea:

```
30 PRINT FLASH 1;"#"
```

Observará otro efecto. Por supuesto, no es necesario convertir toda la pantalla en un carácter fugaz. De hecho, la orden FLASH es mucho más efectiva si se utiliza selectivamente. Pruebe el siguiente programa:

USO SELECTIVO DE **FLASH**



Color transparente y contraste

Aunque el Spectrum sólo tiene ocho colores, el lector puede tropezar con expresiones en las que aparecen los códigos de color 8 ó 9. El color número 8 genera color "transparente". Si se introduce PAPER 8 en una sentencia PRINT, el color del fondo permanece igual que antes.

El color número 9 produce el efecto contrario. Si se

han hecho pruebas con mezclas de colores se habrá observado que algunos colores no se mezclan bien.

En algunas combinaciones el texto se confunde con el fondo y es totalmente ilegible. INK 9 permite obtener texto legible de entrada. Imprime los caracteres en negro y blanco, dependiendo de cual de estos colores contraste más con el fondo:

PROGRAMA **INK 9**

```
  10 FOR p=0 TO 7
  20 INK 9: PAPER p: CLS
  30 PRINT AT 8,5;"INK 9 AUTOMAT
ICALLY"
  40 PRINT AT 10,5;"CONTRASTS IN
K COLOUR"
  50 PRINT AT 12,5;"WITH PAPER C
OLOUR"
  60 PAUSE 50
  70 NEXT p
```

Se imprime un mensaje sobre cada uno de los siete fondos posibles. El color del texto lo elige el ordenador de forma que resalte todo lo posible sobre el color de fondo que hayamos elegido para la pantalla. Sobre fondo negro, rojo, azul y rosa se imprimen caracteres blancos pero si se trata de fondo verde, azul (cyan), amarillo o blanco, el texto será negro.

Sobreimpresión sin borrado

Hasta ahora, cada vez que se imprime sobre un texto en la pantalla, el texto original desaparece bajo los nuevos caracteres. De todas formas, no es necesario que suceda esto. La palabra clave OVER, con la que ya trabajamos en las dos páginas anteriores, es aplicable también a caracteres, además de a gráficos.

Con OVER pueden unirse caracteres. Por ejemplo, muchas palabras alemanas llevan diéresis sobre las letras a, o y u.

El Spectrum puede generar acentos, diéresis y símbolos dobles con programas capaces de reimprimir textos, como el expuesto en la primera pantalla de la figura. El mismo principio es aplicable al subrayado de palabras sobre la pantalla (pantalla central de la figura). Los caracteres del subrayado se colocan exactamente en las posiciones ocupadas por los caracteres que se desean resaltar. Pero no borra las palabras, sino que se añade a ellas. En la última pantalla de la figura contigua puede ver el resultado. Observe los resultados al eliminar la línea 20:

PROGRAMAS DE SOBREIMPRESIÓN

```
  10 PRINT AT 10,10;"Koln"
  20 PRINT OVER 1;AT 10,11;"""""
  30 PRINT AT 12,10;"Societe"
  40 PRINT OVER 1;AT 12,14;"'"
  50 PRINT OVER 1;AT 12,16;"'"
  60 PRINT AT 14,10;"000000000"
  70 PRINT OVER 1;AT 14,10;"-----
-----"


0 OK, 0:1
```

```
  10 PAPER 0: BORDER 0: INK 7: C
LS
  20 OVER 1
  30 PRINT AT 8,8;"OVER LETS YOU
"
  40 PRINT AT 10,10;"UNDERLINE"
  50 PRINT AT 10,10;"_________"
  60 PRINT AT 12,6;"WORDS ON THE
SCREEN"
```

```
OVER LETS YOU
UNDERLINE
WORDS ON THE SCREEN


0 OK, 60:1
```

La orden OVER permite transformar letras en caracteres gráficos, crear distintos alfabetos y, generar un carácter sobre una línea de un fondo cualquiera de la pantalla en el caso de juegos y programas gráficos.

SONIDO, NOTAS Y MÚSICA

El Spectrum es capaz de producir un amplio espectro de sonidos bajo el control de una única palabra clave: BEEP. BEEP va siempre acompañado de dos variables -d (duración del sonido) y p (tono). Por ejemplo:

```
BEEP 1,0
```

produce un pitido de un segundo de duración en un tono C medio. El tono de los sonidos es relativo a C medio —sobre C medio, p es positivo y por debajo p es negativo. El rango de valores de tono del Spectrum se extiende desde -60 a +69. Un incremento de una unidad en p, aumenta el tono del sonido en un semitono. Un semitono es la diferencia de tono entre, por ejemplo, C y C. Aunque de representa la longitud del sonido en segundos, su valor no se restringe a números enteros; las fracciones de segundo son también admisibles.

Atendiendo a la tabla de valores de p, se observará que se puede aumentar el tono de un sonido en una octava sumando 12 a p. El siguiente programa recorre todo el rango de notas del Spectrum. Completa la escala en unos 15 segundos:

ESCALA DE TONOS

```
  10 PRINT AT 10,5;"************
********"
  20 PRINT AT 16,5;"************
********"
  30 FOR p=-60 TO 69
  40 PRINT AT 12,9;"PITCH NUMBER
 "
  50 PRINT AT 14,12;" ";p;" "
  60 BEEP 0.1,p
  70 PAUSE 10
  80 NEXT p


0 OK, 0:1
```

Rango de valores del tono

El rango de valores del tono se extiende desde -60 a + 69. Se muestran los valores de seis octavos agrupados en torno a C(0) medio:

Nota	Valor del tono					
G#,Ab	-16	-4	8	20	32	44
G	-17	-5	7	19	31	43
F#,Gb	-18	-6	6	18	30	42
F	-19	-7	5	17	29	41
E	-20	-8	4	16	28	40
D#,Eb	-21	-9	3	15	27	39
D	-22	-10	2	14	26	38
C#,Db	-23	-11	1	13	25	37
C	-24	-12	0	12	24	36
B	-25	-13	-1	11	23	35
A#,Bb	-26	-14	-2	10	22	34
A	-27	-15	-3	9	21	33

El bucle FOR... NEXT de las líneas 30 a 80 recorre todos los posibles valores de p, desde -60 a 69, haciendo sonar cada nota durante una décima de segundo; las líneas 10 y 20 trazan un marco alrededor del valor de p que imprime la sentencia 50.

Medición de la velocidad del Spectrum con sonido

El programa siguiente utiliza BEEP como señal. Para averiguar la velocidad de cálculo del Spectrum, puede Vd. escribir un programa que realice una serie de operaciones y medir el tiempo que invierte hasta completar la serie. Para una medida aproximada, basta con un reloj y la pantalla. Se utiliza BEEP para marcar el principio y final del programa; y no tener que mirar al reloj y a la pantalla a la vez:

PROGRAMA MEDIDOR DE VELOCIDAD

```
  10 PRINT AT 6,4;"SPECTRUM SPEE
D TEST"
  20 PAUSE 100
  30 LET t=0: BEEP 0.1,25
  40 PRINT AT 9,4;"calculation s
ubtotal"
  50 FOR c=1 TO 1000
  60 LET t=t+c
  70 PRINT AT 12,11;t
  80 NEXT c
  90 BEEP 0.1,30
 100 PRINT AT 9,4;"     FINAL TOT
AL
     "


0 OK, 0:1
```

```
SPECTRUM SPEED TEST

FINAL TOTAL

500500


0 OK, 100:1
```

Una vez que aparece el título del programa, comienza la toma de tiempos.

Un mensaje en la pantalla junto con un pitido de 1/10 de segundo indica cuándo comienza la suma de todos los números desde 1 a 1000. Unos cuantos segundos más tarde, se oye un segundo pitido, de nuevo 1/10 de segundo de duración, pero de un tono más alto; este sonido marca el final del período de cronometraje. En este tiempo el Spectrum ha realizado ¡1000 operaciones y 1000 impresiones!. Dividiendo el tiempo total entre 1000, se obtiene el tiempo que el Spectrum invierte en cada operación y en su impresión.

Programación de melodías sencillas

Con BEEP pueden conseguirse sencillas melodías. Para empezar, se muestra una corta secuencia de notas:

SECUENCIA DE NOTAS SENCILLA

```
10 BEEP 0.5,0
20 BEEP 0.5,4
30 BEEP 0.5,7
40 BEEP 0.5,12
50 BEEP 0.5,7
60 BEEP 0.5,4
70 BEEP 0.5,0

0 OK, 0:1
```

Como cada pitido dura 0,5 segundos, podría ahorrarse algo de tiempo si se usase una variable t, inicializada a 0,5 al principio del programa. El programa adquiere entonces el siguiente aspecto:

VARIABLE DE DURACION

```
 5 LET t=0.5
10 BEEP t,0
20 BEEP t,4
30 BEEP t,7
40 BEEP t,12
50 BEEP t,7
60 BEEP t,4
70 BEEP t,0

0 OK, 0:1
```

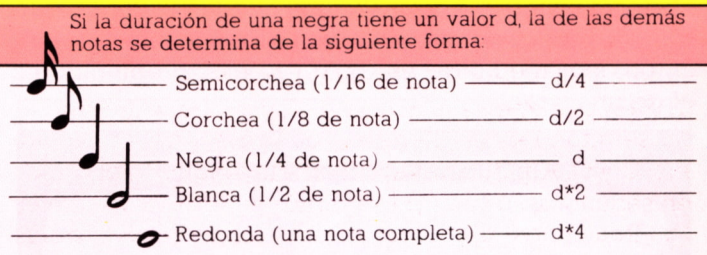
La ventaja de este proceder es que el valor de t puede modificarse, si se desea, al principio del programa.

Música con el Spectrum

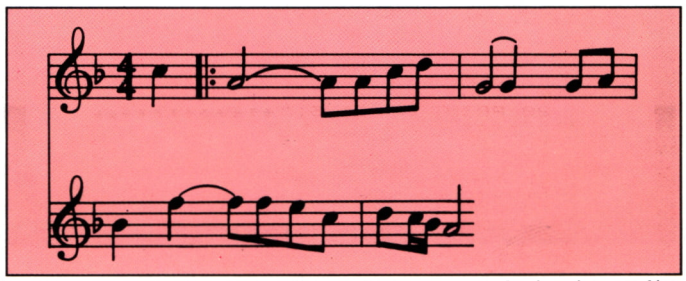
Para escribir música en el Spectrum, hay que resolver el problema de la duración de las distintas notas. Si se denomina d a la duración de una negra, entonces existe la siguiente relación con la duración de las otras notas:

DURACION DE LAS NOTAS

Si la duración de una negra tiene un valor d, la de las demás notas se determina de la siguiente forma:



Un trozo de música escrita puede convertirse en un programa. He aquí unas cuantas notas de una melodía que probablemente resultará familiar al interpretarla con el Spectrum:



Ahora se trata de traducir las notas; el siguiente listado inicializa d a 0.5; una vez se ha introducido el programa, puede variarse la velocidad cambiando este valor. Se apreciará que el tono permanece constante:

```
 5 LET d=0.5
10 BEEP d,12
20 BEEP 2.5*d,9
30 BEEP d/2,9
40 BEEP d/2,12
50 BEEP d/2,14
60 BEEP 3*d,7
70 BEEP d/2,7
80 BEEP d/2,9
90 BEEP d,10
100 BEEP 1.5*d,17
110 BEEP d/2,17
120 BEEP d/2,16
130 BEEP d/2,12
140 BEEP d/2,14
150 BEEP d/4,12
160 BEEP d/4,10
170 BEEP 2*d,9

0 OK, 0:1
```

EFECTOS ESPECIALES CON SONIDO

Hasta el momento se ha usado la palabra BEEP para producir efectos especiales. Pero es posible que, en algún momento, se desee generar sonidos especiales que den más realismo a los programas. Los sonidos amenazantes añaden una nueva dimensión a muchos programas.

Bucles de sonido

El efecto sonoro más sencillo que puede producirse utiliza dos pitidos de distinto tono, formando un bucle con ambos por medio de un GOTO. El efecto obtenido es el de una sirena:

BUCLE DE SONIDO **GOTO**

```
10 BEEP 0.25,9
20 BEEP 0.25,7
30 GO TO 10

0 OK, 0:1
```

Este programa genera una nota alta durante un cuarto de segundo, seguido de una baja durante el mismo período de tiempo. Las dos sentencias BEEP pueden escribirse en la misma línea:

```
10 BEEP 0.25, 9: BEEP 0.25,7
20 GOTO 10
```

Como el programa es en sí un bucle infinito, habrá que accionar CAPS SHIFT y BREAK para interrumpir su ejecución. Si se convierte el bucle en un bucle FOR... NEXT, puede incorporarse a un programa que lo utilice como efecto sonoro.

Alterar la duración de un bucle de sonido

Un sonido puede variarse radicalmente acortando la duración de los elementos que lo componen. Si se modifica el valor de t en la línea 10 de los siguientes programas, se apreciará el efecto de acortar una secuencia de sonido. Este tipo de sonidos se utiliza en rutinas de juegos para asustar al jugador. Cuanto más corta es la duración de los pitidos, más apremiante es el sonido obtenido:

BUCLE DE EFECTOS SONOROS

```
10 LET t=0.1
20 FOR n=1 TO 20
30 BEEP t,50  BEEP t,45  BEEP
t,40  BEEP t,45
40 NEXT n
```

```
10 LET t=0.1
20 BEEP t,-25
30 BEEP t,-24
40 BEEP t,-23
50 BEEP t,-22
60 BEEP t,-21
70 GO TO 20

0 OK, 0:1
```

```
10 FOR n=1 TO 15
20 FOR p=60 TO 30 STEP -2
30 BEEP 0.005,p
40 NEXT p
50 NEXT n

0 OK, 0:1
```

El sonido programado en la tercera pantalla es un ejemplo de lo que sucede cuando el pitido es muy corto. El programa contiene únicamente una sentencia BEEP pero introduce una nueva técnica —el retroceso en un bucle. El bucle comienza con p=60 y disminuye el valor de p en cada ciclo en dos unidades, hasta alcanzar el valor p=30. Cada nota dura sólo 0.005 segundos. Si se acorta más la nota, sonará como un chasquido. Con estos elementos, no puede producirse una gran variedad de sonidos. De todas formas, si se usan los sonidos de tono más bajo, puede obtenerse otro efecto diferente. El programa siguiente produce un sonido similar al de una máquina:

BUCLE DE TONO BAJO

```
10 FOR n=1 TO 100
20 FOR p=-30 TO -25
30 BEEP 0.01,p
40 NEXT p
50 NEXT n


0 OK, 0:1
```

Sonidos no predecibles

Hasta el momento, tiene una idea bastante aproximada del sonido que un programa va a producir antes de su ejecución. Pruebe a continuación el programa (necesitará la palabra RND, sobre la tecla T):

SONIDO ALEATORIO

```
10 LET p=INT (RND*60)
20 BEEP 0.01,p
30 GO TO 10


0 OK, 0:1
```

Este programa permite al ordenador asignar valores al tono de forma aleatoria. (El uso de RND se explica en las páginas 48-49).

Efectos sonoros sincronizados

Es sencillo generar sonidos aislados, pero incorporar efectos sonoros a un programa satisfactoriamente, es más complejo. El truco está en obtener el sonido con la duración correcta en la parte apropiada de un programa. El siguiente programa ilustra cómo se consigue el efecto deseado. Genera un carácter gráfico sencillo que se mueve de un sitio a otro de la pantalla, y produce un efecto sonoro de acuerdo con el movimiento:

SINCRONIZACIÓN DE MOVIMIENTO Y SONIDO

```
  10 BORDER 2: PAPER 6: INK 0: C
LS
  20 FOR c=0 TO 28
  30 LET r=11
  40 BEEP 0.02,12
  50 PRINT AT r,c+1;"▀"
  60 PRINT AT r+1,c+1;"▄"
  70 PRINT AT r+2,c;"▝▘ ▗▖"
  80 BEEP 0.2,0
  90 PRINT AT r,c+1;" "
 100 PRINT AT r+1,c+1;" "
 110 PRINT AT r+2,c;"  "
 120 BEEP 0.02,-5
 130 NEXT c


K
```

En vez de utilizar gráficos definidos por el usuario, el programa usa los caracteres gráficos proporcionados por el sistema. El carácter se compone de ocho símbolos, impresos sobre tres líneas.

En condiciones normales, el carácter se imprime y, después de una breve retención, se borra y se reimprime en la siguiente posición de la pantalla.

El retraso garantiza la permanencia del carácter sobre la pantalla más tiempo del que invierte fuera, minimizando las vibraciones. En el ejemplo se utiliza el efecto sonoro como retención. Si en cada ciclo de bucle FOR... NEXT (líneas 20 a 130) se generase una nota, el sonido resultante sería desarreglado y tartamudeante, debido a los largos silencios existentes. Al dividir los efectos sonoros en múltiples sentencias BEEP evita estos resultados. También pueden generarse sonidos más complejos.

Las longitudes de los sonidos se escogen cuidadosamente, de forma que la retención existente entre el borrado de un carácter y la impresión del siguiente sera mínimo. Debido a que la sentencia BEEP interrumpe un programa durante el tiempo de generación de sonido, es necesario imprimir el carácter sobre la pantalla primero y producir el sonido después.

PROGRAMACION DE PUNTOS DE DECISIÓN

Desde la página 26, se ha aplicado el concepto de bucle. Para efectuar un cálculo o una escritura sobre la pantalla 10 veces, se escribe:

```
FOR A= 1 TO 10...
NEXT A
```

Pero existe otra forma de conseguir esto con una sentencia IF... THEN. Se desea, por ejemplo, imprimir los números del 1 al 10, junto con sus cuadrados, y ambos en una tabla. A continuación se muestra cómo se haría un bucle con FOR... NEXT y con IF... THEN;

BUCLE **FOR... NEXT**

```
  10 BORDER 1: PAPER 6: INK 2: C
LS
  20 PRINT AT 4,6;"A";AT 4,14;"A
↑2";AT 4,24;"A↑3";AT 5,0;"
"
  30 FOR n=1 TO 10
  40 PRINT
  50 BEEP 0.1,40
  60 PRINT AT n+6,6;n;AT n+6,14;
n↑2;AT n+6,24;n↑3
  70 NEXT n


0 OK, 0:1
```

```
A          A↑2          A↑3

1          1            1
2          4            8
3          9            27
4          16           64
5          25           125
6          36           216
7          49           343
8          64           512
9          81           729
10         100          1000

0 OK, 70:1
```

En el programa con sentencia IF... THEN, la línea 10 establece el color de la pantalla y la 30 imprime la cabecera de la tabla de la misma forma que antes. La sentencia 40 es la primera del bucle —aumenta n en una unidad en cada ciclo. La línea 70 es la misma sentencia de impresión utilizada en el programa con el bucle FOR... NEXT. En la línea 80 el ordenador debe realizar una decisión en función del valor de n. El símbolo < representa "menor que". Entonces, si el n es menor que 10, el ordenador ejecuta el nuevo programa desde la línea 40. El sistema imprime A, A12, A13, hasta que n es mayor o igual que 10.

BUCLE **IF... THEN**

```
  10 BORDER 1: PAPER 6: INK 2: C
LS
  20 LET n=0
  30 PRINT AT 4,6;"A";AT 4,14;"A
↑2";AT 4,24;"A↑3";AT 5,0;"
"
  40 LET n=n+1
  50 PRINT
  60 BEEP 0.1,40
  70 PRINT AT n+6,6;n;AT n+6,14;
n↑2;AT n+6,24;n↑3
  80 IF n<10 THEN GO TO 40


0 OK, 0:1
```

¿Por qué usar el bucle IF... THEN?

El lector se preguntará qué objeto tiene usar el bucle IF... THEN si con FOR... NEXT se obtienen los mismos resultados. La utilidad del bucle IF... THEN reside en que el ordenador puede responder a cualquier información introducida durante la ejecución del programa. El ejemplo siguiente muestra esta propiedad:

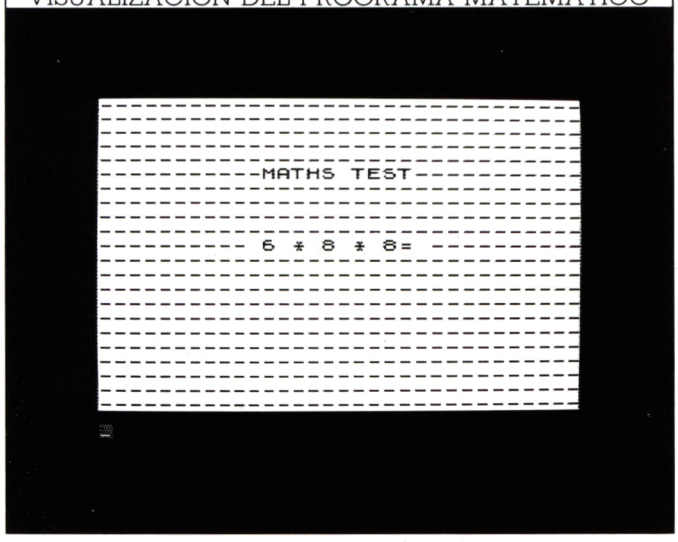
PROGRAMA EXAMEN DE MATEMÁTICAS

```
  10 BORDER 0: CLS
  20 FOR r=0 TO 21
  30 FOR c=0 TO 31
  40 PRINT AT r,c;"-"
  50 NEXT c: NEXT r
  60 PRINT AT 5,11;"MATHS TEST"
  70 LET x=INT (RND*10): LET y=I
NT (RND*10): LET z=INT (RND*10)
  80 PRINT AT 10,10;" ";x;" * ";
y;" * ";z;"= ": INPUT a
  90 IF a=x*y*z THEN GO TO 130
 100 BEEP 0.5,40: PRINT AT 10,10
;"---Wrong------": PAUSE 50
 110 PRINT AT 10,10;"-Try again-
": PAUSE 50
 120 GO TO 80
 130 BEEP 0.5,50: PRINT AT 10,10
;"--Correct---": PAUSE 50
 140 GO TO 70


0 OK, 0:1
```

Cada vez que el ordenador propone un problema y espera una respuesta, se enfrenta a dos posibles acciones diferentes. Si se teclea la respuesta correcta, la sentencia IF... THEN de la línea 90 obliga a que el control pase, no a la línea 100, sino a la 130 , donde se imprime un mensaje y se plantea un nuevo problema. Si la respuesta es incorrecta, la ejecución continúa con la sentencia 100, y se ejecuta la rutina de "respuesta falsa".

Es importante recordar que el programa debe contener alguna sentencia que impida que la ejecución de la rutina de "respuesta falsa" continúe con la de rutina de "respuesta correcta". En este caso es la línea 120 la que resuelve el problema:

VISUALIZACIÓN DEL PROGRAMA MATEMÁTICO



Operación de gráficos IF...THEN

Puede utilizarse IF... THEN con las órdenes para gráficos y convertir el Spectrum en un sistema de dibujo. Todo lo que hay que hacer es programar el ordenador para que responda a entradas gráficas.

PROGRAMA GRAFICO IF... THEN

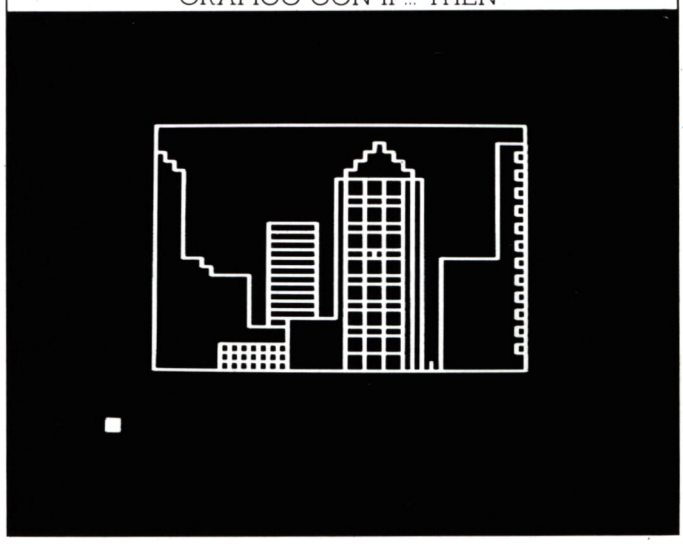
```
  10 INPUT "INK ";i
  20 BORDER 3: PAPER 0: INK i: C
LS
  30 PLOT 25,25
  40 LET x=5
  50 INPUT K$
  60 IF K$="u" THEN DRAW 0,x
  70 IF K$="d" THEN DRAW 0,-x
  80 IF K$="r" THEN DRAW x,0
  90 IF K$="l" THEN DRAW -x,0
 100 GO TO 50


0 OK, 0:1
```

En este programa, las sentencias IF... THEN capacitan al ordenador para decidir qué acción debe realizar. Este programa usa las teclas U, D, L, y R para dibujar líneas abajo, arriba, a izquierda y derecha, desde un punto cerca de la parte baja de la izquierda de la pantalla.

El programa dibuja el punto 25,25 y traza una pequeña línea cada vez que se acciona una de las cuatro letras. Las cuatro líneas IF... THEN hacen que ordenador examine la entrada y decida en qué dirección debe dibujarse la línea. Cada vez que Vd. usa una tecla, recuerde pulsar ENTER para que se reciba la instrucción.

La sentencia 30 indica al ordenador la longitud que ha de tener la línea. Asignando a x el valor 5 se obtienen resultados aceptables; modificando este valor, se cambia la resolución de los dibujos. El color puede cambiarse modificando los números de la línea 10. La pantalla siguiente dará una idea de lo que puede dibujarse.

GRAFICO CON IF... THEN



Selección de la condición adecuada

Evidentemente, este tipo de programa puede extenderse, imponiendo condiciones sobre los valores de más teclas. Pero el escribir largas filas de IF... THEN no es buena táctica de programación.

Sin embargo, cuando utilice IF... THEN, recuerde que hay muchas condiciones que pueden acompañar a la parte IF de la línea. Los ejemplos expuestos sólo utilizan < ó = pero estos forman parte del amplio rango de símbolos que el Spectrum utiliza. El hacer la elección de la condición adecuada no es una tarea fácil, especialmente cuando se trata con caracteres en movimiento.

CONDICIONES IF...THEN

Los símbolos que acompañan a la parte IF de una línea especifican el tipo de decisión que el ordenador debe realizar.

= es igual a	<> no es igual a
> es mayor que	< es menor que
>= es mayor o igual que	<= es menor o igual que

PROGRAMAS NO PREDECIBLES

Aunque habitualmente los ordenadores trabajan con información precisa, ciertas aplicaciones requieren un elemento aleatorio. Por ejemplo, muchos juegos se basan hasta cierto punto en la suerte. Si se desea desencadenar algún suceso en un instante de tiempo desconocido o si hay que tirar dados o lanzar monedas, no puede indicarse de antemano al ordenador el resultado que debe producir en cada momento, si lo que se quiere conservar es el elemento azar.

La palabra clave RND es la que introduce un factor aleatorio en los programas. Ya se ha visto en ejemplos anteriores. RND es la abreviatura de la palabra inglesa RANDOM (azar). Permite generar tantos números aleatorios como se desee, que pueden usarse para formar sentencias aleatorias. RND se utiliza de la siguiente forma:

```
10 A=RND
```

Esta sentencia asigna al número decimal A un valor cualquiera entre 0 y 0.999999999. Use RND en el siguiente programa (imprime números aleatorios).

La línea 120 genera números aleatorios cuyos valores están comprendidos entre 0 y 0.999999999; las líneas

GENERADOR DE NÚMEROS ALEATORIOS

```
 10 PRINT AT 4,4;"Random number
generator"
 20 PRINT AT 5,4;"
"
 30 FOR c=8 TO 21
 40 PRINT AT 8,c;"*"
 50 PRINT AT 12,c;"*"
 60 NEXT c
 70 FOR r=9 TO 11
 80 PRINT AT r,8;"*"
 90 PRINT AT r,21;"*"
100 NEXT r
110 FOR n=1 TO 20
120 PRINT AT 10,10;RND
130 BEEP .1,25
140 PAUSE 25
150 PRINT AT 10,10;"
160 NEXT n
170 STOP


0 OK, 0:1
```

30-100 enmarcan los números con asteriscos. Los números muy pequeños incluyen el símbolo E. A medida que se imprime un nuevo número se borra automáticamente el anterior por el sencillo método de imprimir encima. Cuando aparece algo del tipo de E-4, no se borra automáticamente. Pero de esto se encargan los nueve espacios en blanco de la línea 150.

El Spectrum puede generar números aleatorios dentro del rango usado en este programa. Para generar otros números al azar, hay que manipular la palabra clave RND.

VISUALIZACION DE UN NUMERO ALEATORIO

```
Random number generator

**************
*            *
* 0.2371521  *
*            *
**************
```

Producción de números aleatorios enteros

Si la línea 120 se sustituye por:

```
120 PRINT AT 10,14; INT (RND*10)
```

y se ejecuta el programa de nuevo, se apreciará un cambio inmediato en la pantalla. Los números obtenidos no son números decimales; es más, ni siquiera aparece el punto decimal. El programa genera enteros comprendidos entre 0 y 9 inclusive —un resultado mucho más útil a la hora de programar—. La función INT, comentada en la página 27, redondea el número aleatorio producido al entero más próximo.

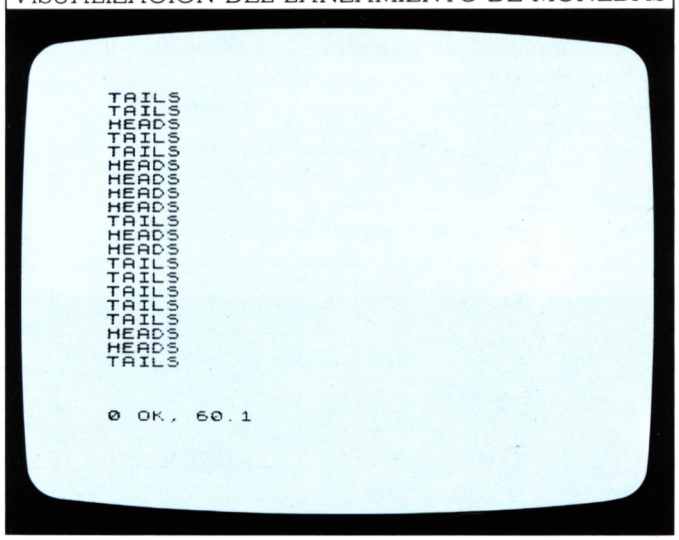
Esta forma de usar RND es muy útil al programar juegos que incluyen un factor aleatorio. Es muy sencillo simular con el Spectrum un lanzamiento de dados o monedas:

LANZAMIENTO DE MONEDAS

```
10 FOR f=1 TO 20
20 LET t=INT (RND*2)+1
30 IF t=1 THEN PRINT "TAILS"
40 IF t=2 THEN PRINT "HEADS"
50 PAUSE 25
60 NEXT f


0 OK, 0:1
```

Como al lanzar una moneda sólo pueden obtenerse dos valores —cara o cruz— la línea 20 genera un número aleatorio que puede ser 1 ó 2. El número 1 representa cruz y el 2 cara. Dos líneas IF... THEN determinan el mensaje para imprimir y luego, la ejecución continúa:

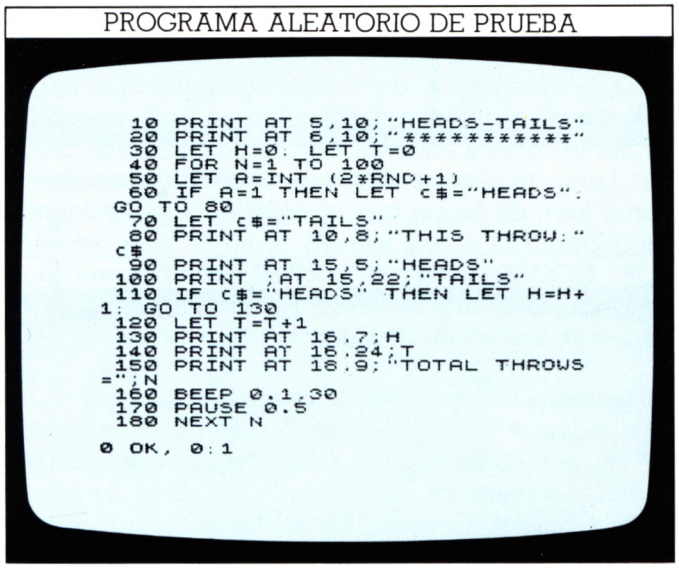
VISUALIZACIÓN DEL LANZAMIENTO DE MONEDAS



Comprobar una secuencia aleatoria

Se puede escribir un programa que muestre el funcionamiento de la orden RND. Si se usa RND para lanzar una moneda electrónica 100 veces, se obtendrán 50 cruces y 50 caras, en cada ejecución. Compruébelo con el siguiente programa:

PROGRAMA ALEATORIO DE PRUEBA

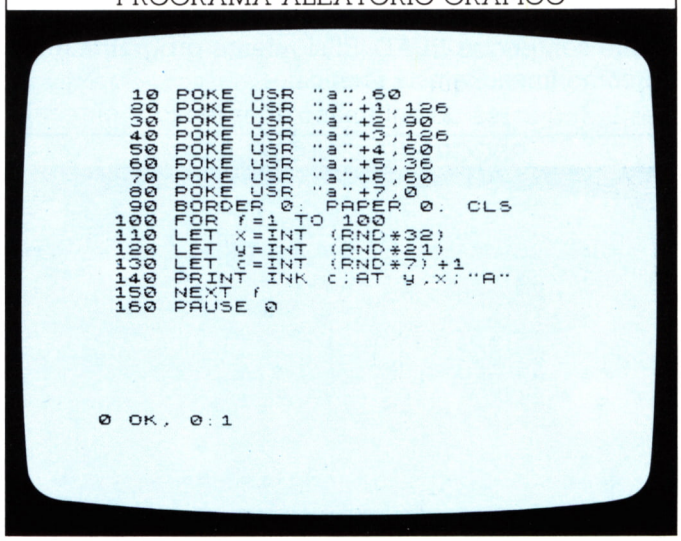


Al ejecutar el programa anterior se comprobará lo cerca que están de 50:50 el número de cruces y caras obtenido en cada ejecución.

Uso de RND en programas gráficos

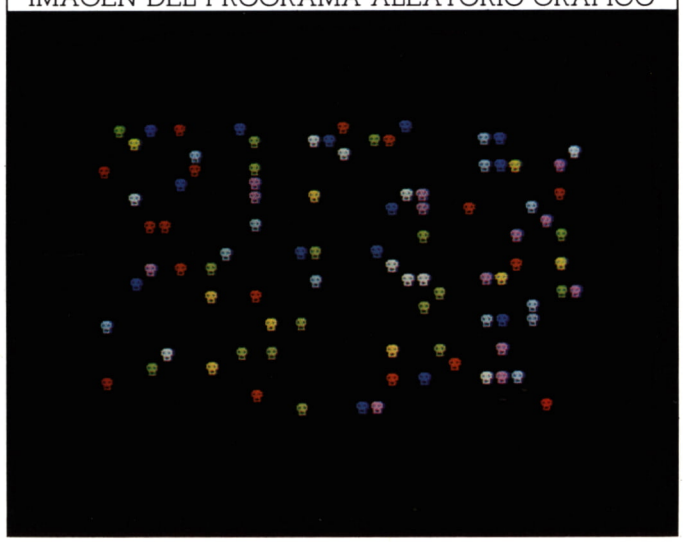
Pueden producirse efectos interesantes si se incorpora RND a programas gráficos, para indicar al ordenador que imprima un carácter en una posición aleatoria de la pantalla. Si se repite esta operación un cierto número de veces mediante un bucle FOR... NEXT, puede construirse una imagen diferente en cada ejecución:

PROGRAMA ALEATORIO GRÁFICO



Las líneas 10-80 definen un carácter que se almacena en la memoria del ordenador y que se visualiza accionando la tecla A y activando el cursor gráfico. La línea 90 configura una pantalla de color negro —tanto el fondo como el borde están a 0. Las sentencias 100 a 150 forman un bucle FOR...NEXT que selecciona una posición y un color aleatorio para el carácter a imprimir. PAUSE 0 (línea 160) detiene la ejecución hasta que se acciona cualquier tecla, impidiendo que aparezca el mensaje de fin de ejecución

IMAGEN DEL PROGRAMA ALEATORIO GRÁFICO



La palabra clave RND puede usarse para que los caracteres del teclado o incluso puntos aleatorios, aparezcan en colores elegidos al azar. Para esto, use la sentencia INK c (donde c es un número aleatorio entre 1 y 7) en la forma indicada en la línea 100 del programa.

BANCO DE DATOS

Los datos que necesita un programa pueden, bien obtenerse en tiempo de ejecución mediante la palabra clave INPUT, o bien formar parte del propio programa. Los datos se definen en sentencias DATA y se leen mediante sentencias READ. El siguiente programa muestra cómo funciona esta técnica:

PROGRAMA CONSTELACIÓN

```
 10 BORDER 1: PAPER 1: INK 7: C
LS
 20 PRINT AT 0,11;"URSA MAJOR"
 30 PAUSE 50
 40 DATA 14,3,10,7,10,11,9,16,5
,24,10,25,13,20
 50 LET n=7
 60 FOR c=1 TO n
 70 READ x,y
 80 BEEP 0.05,25
 90 PRINT AT x,y;"*"
100 PAUSE 20
110 NEXT c

0 OK, 0:1
```

Al ejecutar este programa se obtiene en la pantalla un mapa de un grupo de estrellas, la constelación Osa Mayor, también conocida como el Carro:

IMAGEN DE LA CONSTELACIÓN

La información necesaria para generar la imagen se introduce en la línea 40 a partir de 14 coordenadas. La 70 indica al ordenador que debe leer los datos de la línea 40 e interpretarlos como parejas de números x, y. La sentencia 50 aclara que habrá en total siete pares de coordenadas.

El ordenador genera en primer lugar un pulso de sonido corto y, a continuación imprime un asterisco en cada posición x,y, transformando de esta forma la fila de datos en el mapa de estrellas.

A continuación se muestran una serie de cambios en las líneas y el mapa que producen:

```
20 PRINT AT 0,11; "CASSIOPEIA"
40 DATA 8, 3,12,8,9,14,14,18,8,24.
50 LET n=5
```

IMAGEN DE LA CONSTELACIÓN

Cuando se utiliza la sentencia DATA, hay que indicar al ordenador cuántos datos se van a teclear. La línea 50 del programa anterior indica cómo se debe proceder. Limita el número de pares de coordenadas que se van a leer, de forma que el ordenador pare después de imprimir la última estrella. Si no hubiese un bucle FOR... NEXT en la segunda mitad del programa, el ordenador no tendría bastantes datos, y finalizaría la ejecución con un mensaje de error.

Almacenamiento conjunto de cadenas y números

Las cadenas pueden leerse también con sentencias DATA, y también pueden almacenarse conjuntamente cadenas y números, por ejemplo, nombres de amigos y sus números de teléfono o el día de su cumpleaños. Esto presenta algún problema, ya que para leer números y caracteres deben usarse dos sentencias READ distintas. READ y READ$, por ejemplo. Pero si todos los datos se convierten en cadenas se supera esta dificultad.

El programa siguiente produce una lista de números de teléfono. Las líneas 10-50 cargan los nombres y números de teléfono; las sentencias 60-80 exhiben el título del programa y ofrecen un conjunto de funciones:

PROGRAMA LISTÍN TELEFÓNICO

```
  10 DATA "C.Barnes","141","R.Bu
ckman","322","J.Hann","166"
  20 DATA "R.Harty","103","S.Ing
le","191","S.Jay","47"
  30 DATA "H.Kelly","33","M.Park
inson","86","M.Philbin","71"
  40 DATA "B.Redhead","122","M.R
odd","100","S.Scott","260"
  50 DATA "M.Stoppard","300","J.
Timpson","90"
  60 PRINT AT 5,4;"PERSONAL TELE
PHONE LIST"
  70 PAUSE 100
  80 PRINT AT 12,2;"COMPLETE LIS
TING....press 1";AT 15,2;"SELECT
IVE LISTING...press 2": INPUT c
  90 CLS
 100 IF c=2 THEN GO TO 160
 110 PRINT : PRINT : PRINT
 120 FOR c=1 TO 14
 130 READ n$,m$
 140 PRINT TAB 6;n$;TAB 18;m$

scroll?
```

```
 150 NEXT c: PAUSE 0: CLS : GO T
O 60
 160 CLS : PRINT AT 9,5;"ENTER I
NITIAL AND NAME"
 170 INPUT e$
 180 LET r$="Name not found"
 190 RESTORE
 200 FOR c=1 TO 14
 210 READ n$,m$
 220 IF n$=e$ THEN LET r$=n$+"
 ..... "+m$
 230 NEXT c
 240 CLS : PRINT AT 10,5;r$
 250 PAUSE 200: CLS
 260 RESTORE : GO TO 60



0 OK, 0:1
```

para imprimir la lista completa, teclee:

1 then ENTER

LISTA TELEFÓNICA COMPLETA

```
C.Barnes      141
R.Buckman     322
J.Hann        166
R.Harty       103
S.Ingle       191
S.Jay         47
H.Kelly       33
M.Parkinson   86
M.Philbin     71
B.Redhead     122
M.Rodd        100
S.Scott       260
M.Stoppard    300
J.Timpson     90
```

Si se teclea la siguiente orden:

2 then ENTER

Se obtiene el número que acompaña a un nombre, el ordenador exhibe de nuevo en menú.

MENU PARA EL PROGRAMA LISTIN TELEFONICO

```
PERSONAL TELEPHONE LIST


COMPLETE LISTING....press 1

SELECTIVE LISTING...press 2
```

Si en la línea 80 se teclea un 2, el programa continúa a lo largo de las líneas 160-260. Primero solicita una inicial y un nombre. Hay que asegurarse de que los datos que se introducen sean idénticos a los de las líneas DATA.

Si el ordenador encuentra que el nombre (e$) que se introduce coincide con uno de los de las sentencias DATA (n$), forma una nueva cadena (r$) con el valor de n$, una línea de puntos y el número de teléfono. Si no encuentra e$, r$ queda inicializado con la cadena "name not found" (nombre no encontrado).

Dado que en la sentencia 220 se desea unir en una sola cadena el nombre, la línea de puntos y el número de teléfono, habrá que tratar este último como una variable cadena m$, en vez de como una variable numérica m. Si se usase m, el programa no funcionaría, ya que no pueden mezclarse variables cadena con variables numéricas.

Las líneas 190 y 260 utilizan una palabra clave nueva: RESTORE. Indica al ordenador que debe volver al principio de las sentencias DATA una vez que ejecute la siguiente sentencia READ. Sin esta nueva palabra clave, la ejecución del programa sólo sería correcta la primera vez. Esto es debido a que el sistema acusaría una carencia de datos; intentaría continuar la búsqueda de datos, pero el programa le indica que en cada ejecución debe leer todos los datos disponibles. RESTORE facilita el que el ordenador comience la búsqueda desde el principio de cada DATA.

Una vez que se aprende a compilar un banco de datos, puede hacerse uno que almacene fechas de cumpleaños, otro para facturas y pagos o para la información relativa a la colección de cintas de video.

FORMAS RAPIDAS DE ALMACENAR CARACTERES

En las páginas 30-31 se vio como se programaban y almacenaban caracteres propios con las palabras clave POKE y USR. Ya habrá observado que la tarea de generar un solo carácter definido por el usuario es bastante larga. Se necesitan ocho líneas para programar una única tecla y para crear un símbolo compuesto de cuatro caracteres se necesitarían 24 líneas de programa. De todas formas, ya se introdujo en las páginas 36-37 una forma abreviada y rápida de producir caracteres definidos por el usuario.

Programar caracteres con READ...DATA

Ya se vio en la página 30 que, para programar un carácter gráfico, hay que teclear el valor numérico de cada línea de la red. He aquí dos programas que generan el mismo carácter. El primero utiliza una línea para almacenar cada valor de la retícula.

AHORRANDO LINEAS CON **FOR... NEXT**

```
 10 POKE USR "a",60
 20 POKE USR "a"+1,126
 30 POKE USR "a"+2,251
 40 POKE USR "a"+3,254
 50 POKE USR "a"+4,248
 60 POKE USR "a"+5,255
 70 POKE USR "a"+6,126
 80 POKE USR "a"+7,60


0 OK, 0:1
```

```
 10 DATA 60,126,251,254,248,255
126,60
 20 FOR f=0 TO 7
 30 READ x
 40 POKE USR "b"+f,x
 50 NEXT f


0 OK, 0:1
```

El segundo programa almacena valores totales en una sentencia DATA y, a continuación lo transforma en caracteres por medio de una linea READ.

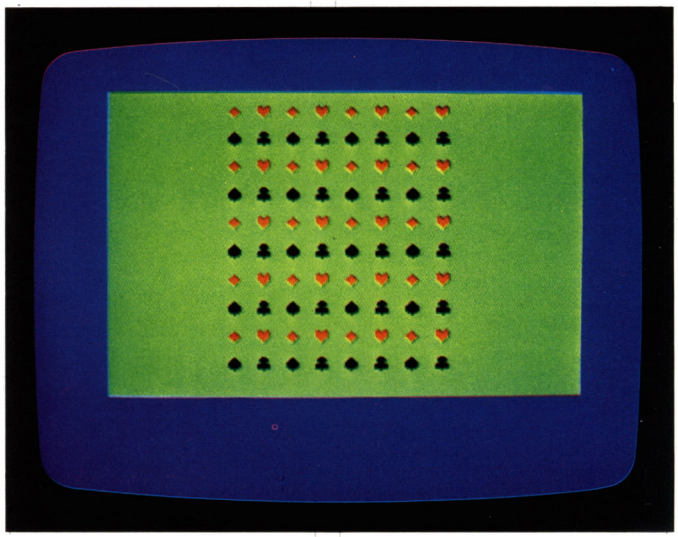
Se observará que READ...DATA disminuye el número de líneas necesarias en tres: de ocho a cinco. Pero aunque esto suponga un considerable ahorro, la gran ventaja de READ... DATA queda patente en la programación de símbolos compuestos de múltiples caracteres o, cuando se desea utilizar varios símbolos en un programa.

Almacenar grupos de caracteres

El siguiente programa reprograma las teclas a, b, c, y d de forma que produzcan los símbolos de los cuatro palos de una baraja de cartas. En cada bucle se genera un símbolo distinto. Si se añaden unas cuantas líneas, se pueden visualizar los símbolos sobre la pantalla:

READ... DATA CON 4 BUCLES

```
 10 DATA 0,8,28,62,127,62,28,8
 20 DATA 34,119,127,127,127,62,
28,8
 30 DATA 8,28,62,127,127,127,62
,8
 40 DATA 28,62,62,28,127,127,12
7,8
 50 FOR f=0 TO 7
 60 READ x
 70 POKE USR "a"+f,x: NEXT f
 80 FOR f=0 TO 7
 90 READ x
100 POKE USR "b"+f,x: NEXT f
110 FOR f=0 TO 7
120 READ x
130 POKE USR "c"+f,x: NEXT f
140 FOR f=0 TO 7
150 READ x
160 POKE USR "d"+f,x: NEXT f


0 OK, 0:1
```



Esta forma de generar los cuatro símbolos utiliza en total 16 líneas, es decir, ocho líneas menos que el laborioso método que utiliza ocho sentencias POKE... USR por cada símbolo. Puede ahorrarse aún más espacio leyendo los datos en un único bucle FOR... NEXT:

READ... DATA CON UN BUCLE

```
  10 DATA 0,34,8,28,8,119,28,62
  20 DATA 28,127,62,62,62,127,12
7,28
  30 DATA 127,127,127,127,62,62,
127,127
  40 DATA 28,28,62,127,8,8,8,8
  50 FOR f=0 TO 7
  60 READ w,x,y,z
  70 POKE USR "a"+f,w
  80 POKE USR "b"+f,x
  90 POKE USR "c"+f,y
 100 POKE USR "d"+f,z
 110 NEXT f


0 OK, 0:1
```

A primera vista, parece que se han cambiado todas las sentencias DATA. Pero lo que realmente ha sucedido es que se ha modificado el orden de los números en las sentencias. Las sentencias DATA ocupan las líneas 10-40. La línea 60 ordena al ordenador que lea estos datos en grupos de cuatro números, w, x, y, z; cada uno de estos números reprograma una tecla. Es decir, los números que reprograman la letra A son el primero, quinto, noveno, etc. Las sentencias DATA se distribuyen:

```
DATA a, b, c, d, a, b, c, d, a, b, c, d, ...
```

Para usar este método con caracteres propios, basta con obtener los valores totales de la red, como en el diagrama de la página 30 e introducirlos en líneas DATA en el orden correcto. Ahora se ahorran trece líneas de programa en vez de ocho.

Cómo usar números binarios con gráficos

A no ser que se disponga de una calculadora, el programar caracteres gráficos resulta toda una prueba de capacidad de cálculo. Los códigos numéricos de cada columna de cuadrados no son precisamente muy manejables y es fácil cometer errores. Pero el Spectrum permite al usuario introducirlos de otra forma, usando la palabra clave BIN, abreviatura de BINario.

Todos los ordenadores utilizan un sistema de pulsos binarios para manipular la información. La palabra "binario" quiere decir que hay sólo dos tipos de pulsos "encendido" ó 1 y "apagado" ó 0. Los programas no son más que una colección de pulsos electrónicos. Cada número que se teclea en el Spectrum se transforma en una secuencia de pulsos "encendido" y "apagado" pero, para facilitar la tarea al usuario se diseña el ordenador de forma que acepte números "usuales".

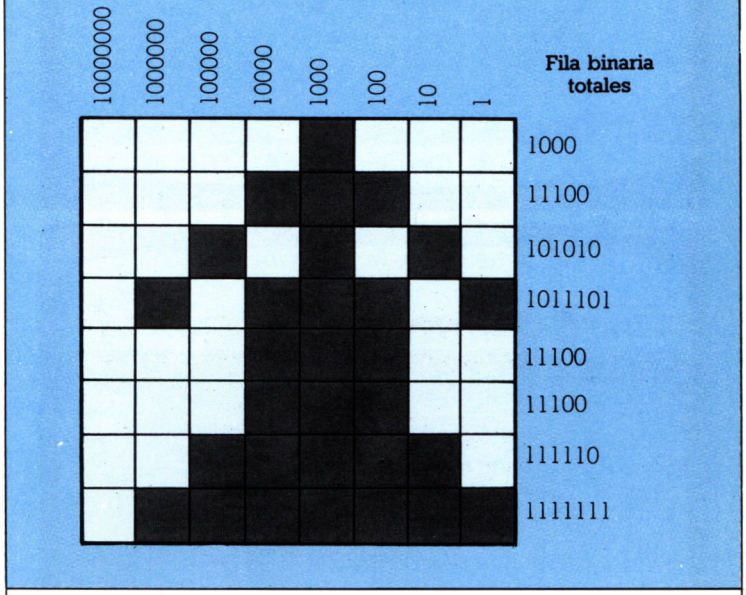
La palabra clave BIN permite introducir números binarios directamente. BIN precede siempre a un número expresado en múltiplos de dos en vez de en múltiplos de diez. Reinicialice el ordenador y observe lo que sucede al teclear:

```
PRINT BIN 10
PRINT BIN 100
PRINT BIN 1011
```

En vez de exhibir 10, 100, 1011, el sistema imprime 2, 4, y 11 en la pantalla. El ordenador ha transformado números binarios en números "usuales". La orden BIN es muy útil a la hora de reprogramar teclas. Recordará que los números situados sobre las columnas de la retícula no eran consecutivos. Cada número es el doble del situado a su derecha.

El ordenador otorga a cada columna un código binario, 0, 10, 100, 1000, etc. de forma que pueden utilizarse números binarios en la confección de caracteres. La siguiente red es binaria.

USO DE RED BINARIA



Si se imagina que cada cuadrado gris tiene un valor de 1 y cada cuadrado vacío un valor de 0, pueden teclearse los valores totales de cada fila como secuencias de unos y ceros, por medio de una sentencia BIN o bien, de una serie de sentencias BIN dentro de órdenes de impresión de valor decimal de cada valor total. La gran ventaja de BIN es que no hay que realizar sumas. Basta con teclear lo que se ve, contando negro como 1 y blanco como O.

GRAFICOS EN COLOR AVANZADOS

Una vez que se han probado algunos gráficos en color en las páginas 36-37, pueden reunirse todos los elementos relativos a gráficos, color y movimiento en un programa y ver cómo interaccionan. El programa de estas dos páginas realiza un dibujo complejo. Al trabajar con este programa Vd. mismo puede llegar a crear uno nuevo.

Creación de paisaje

El primer paso en la producción de una imagen es programarla en blanco y negro. El listado siguiente efectúa esta tarea, utilizando el sistema DATA explicado en las páginas anteriores, junto con la palabra clave PLOT y DRAW para rellenar dos "pirámides":

PROGRAMA ANTES DE COLOREAR

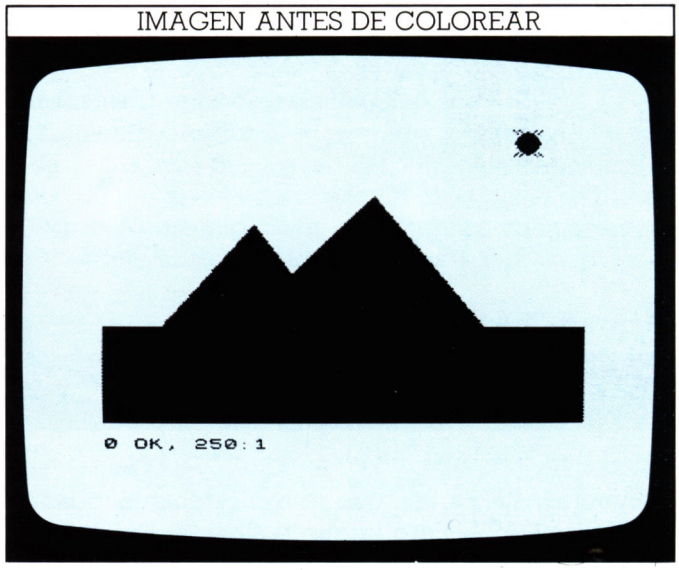
```
  10 DATA 144,9,127,254,73,146,6
3,252
  20 DATA 39,228,63,252,15,240,3
1,248
  30 DATA 31,248,15,240,63,252,3
9,228
  40 DATA 63,252,73,146,127,254,
144,9
  50 FOR n=0 TO 7
  60 READ p,q,r,s
  70 POKE USR "a"+n,p
  80 POKE USR "b"+n,q
  90 POKE USR "c"+n,r
 100 POKE USR "d"+n,s
 110 NEXT n
 120 FOR r=15 TO 21
 130 FOR c=0 TO 31
 140 PRINT AT r,c;"■"
 150 NEXT c: NEXT r
 160 FOR x=24 TO 128
 170 PLOT 80,112
 180 DRAW x-80,-65

scroll?
```

```
 190 NEXT x
 200 FOR x=72 TO 200
 210 PLOT 144,128
 220 DRAW x-144,-72
 230 NEXT x
 240 PRINT AT 1,27;"AB"
 250 PRINT AT 2,27;"CD"









0 OK, 0:1
```

Las líneas DATA (10-40) definen cuatro caracteres que las sentencias 240 a 250 imprimen conjuntamente, dibujando el sol. Las líneas 50 a 110 transforman la información de las sentencias DATA en caracteres definidos por el usuario, controlados mediante las teclas a, b, c y d. Las sentencias 120 a 150 trazan siete líneas con el carácter gráfico de la tecla 8. Las líneas 160 a 230 dibujan dos pirámides a partir de triángulos.

Una vez que se ha tecleado el programa, ejecútelo para comprobar que es correcto. Si obtiene la siguiente imagen, puede pasar a la próxima etapa:

IMAGEN ANTES DE COLOREAR



Programación de perspectiva y color

El añadir el color es una cuestión sencilla:

```
  5 BORDER 0:PAPER 1:CLS
140 PRINT INK 2;AT r,c,"■"
235 INK 6
```

Al ejecutar el programa, los bordes serán de color negro, el fondo azul, el primer plano rojo y el sol amarillo.

Puede añadir unas líneas que den al paisaje una sensación de profundidad. Estas líneas parecerán caminos paralelos, que desaparecen en la distancia. Para esto se dibujan segmentos entre dos filas horizontales de coordenadas, con las puntas superiores más próximas que las inferiores. Teclee las líneas siguientes:

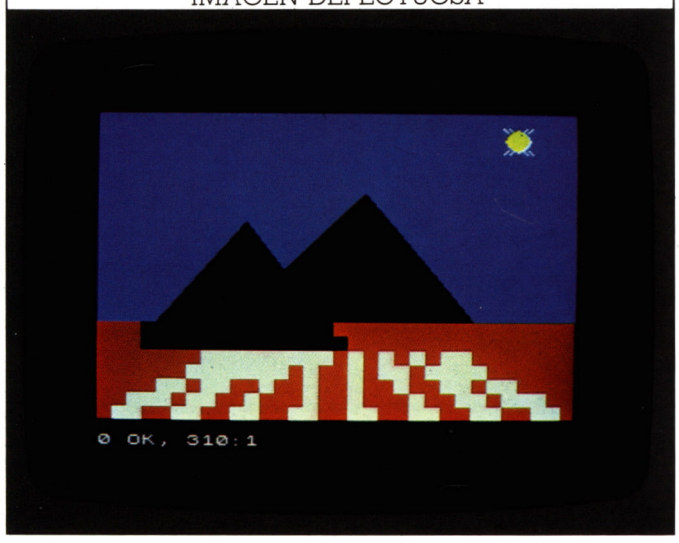
```
260 LET w=8: INK 7
270 FOR x=72 TO 184 STEP 16
280 PLOT x,38
290 DRAW w-x,-38
300 LET w=w+34
310 NEXT x
```

La línea 270 da la abscisa del extremo superior de cada línea. STEP 16 aumenta x en 16 unidades cada vez en vez en una unidad.

Entonces se desplaza el cursor gráfico a la parte superior de la línea con PLOT x,38. La línea 290 traza cada

recta desde ese punto hasta la parte inferior de la pantalla. Para que la coordenada x del extremo inferior de la recta avance a intervalos mayores que el superior, la línea 300 incrementa w en 34 unidades. Una vez hecho esto, ejecute el programa:

IMAGEN DEFECTUOSA



Apreciará que los resultados son un tanto extraños. Las rectas trazadas son realmente grandes cuadrados. ¿Qué ha sucedido?

Dibujar líneas en color

La respuesta es que, si bien los gráficos del Spectrum utilizan 256×170 pixels, la resolución del color y texto es de 32×22. A medida que se trazan las líneas, se modifica el color del cuadrado por el que se pasa. Pero si atiende a la mitad superior de la imagen observará que las líneas que componen las pirámides no generan gráficos defectuosos. Es necesario efectuar una nueva modificación para obtener el siguiente gráfico correcto:

IMAGEN CORREGIDA

El problema reside en que las líneas pueden trazarse con éxito sobre fondo dibujado con la palabra clave PAPER pero no sobre zonas coloreadas con INK.

Hay que reescribir el programa para colorear el suelo en rojo con PAPER —no con INK. Modifique la línea 140:

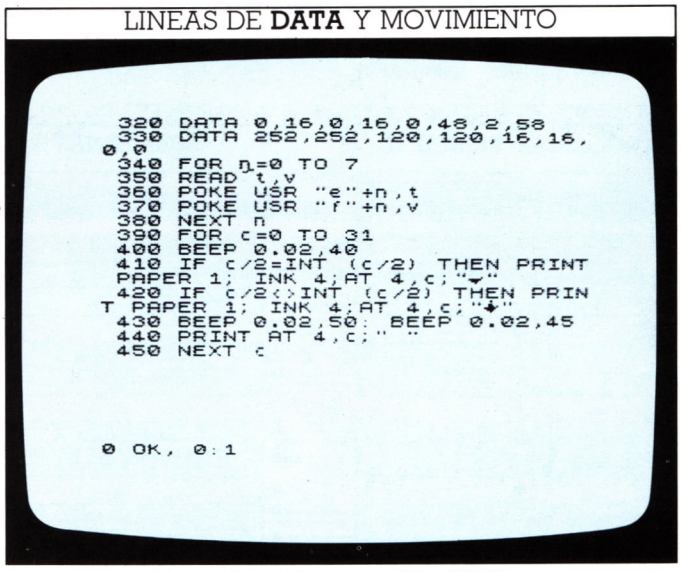
```
140 PRINT PAPER 2;AT r,c;" "
```

Ahora se obtienen las líneas perfectamente.

Añadir un carácter móvil

Como toque final, puede añadir un pájaro volando a través de la pantalla. El pájaro puede formarse alternando dos caracteres definidos por el usuario, para simular el batir de alas:

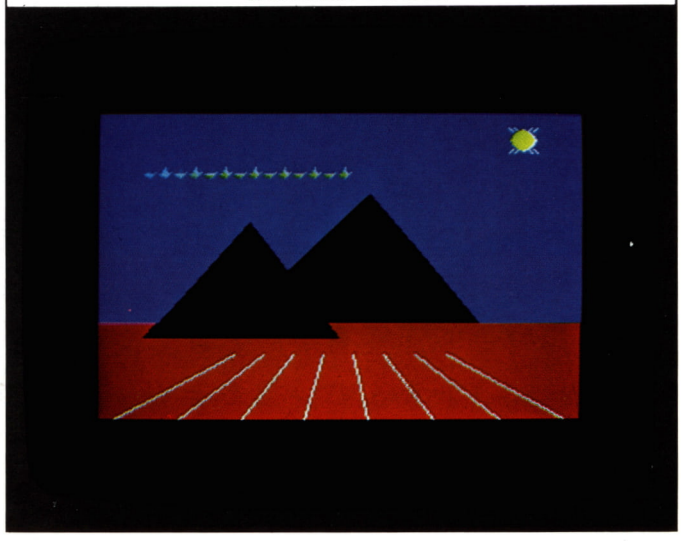
LINEAS DE **DATA** Y MOVIMIENTO



```
 320 DATA 0,16,0,16,0,48,2,58
 330 DATA 252,252,120,120,16,16,
0,0
 340 FOR n=0 TO 7
 350 READ t,v
 360 POKE USR "e"+n,t
 370 POKE USR "f"+n,v
 380 NEXT n
 390 FOR c=0 TO 31
 400 BEEP 0.02,40
 410 IF c/2=INT (c/2) THEN PRINT
 PAPER 1; INK 4;AT 4,c;"e"
 420 IF c/2<>INT (c/2) THEN PRIN
T PAPER 1; INK 4;AT 4,c;"f"
 430 BEEP 0.02,50: BEEP 0.02,45
 440 PRINT AT 4,c;" "
 450 NEXT c


0 OK, 0:1
```

Las líneas 410 y 420 utilizan un IF...THEN para comprobar si la columna que ocupa el pájaro, c, es par o impar. Si es par, se imprime sólo el cuerpo del pájaro y, si es impar, se añade un ala extendida. La imagen completa:

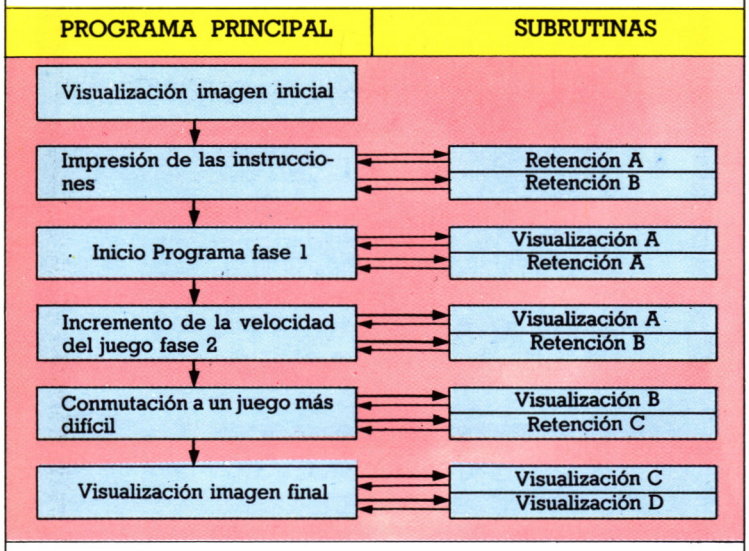
IMAGEN FINAL



ESCRITURA DE SUBRUTINAS

Muchas veces se deseará usar un grupo de líneas de un programa de nuevo, bien para efectuar la misma operación, bien para visualizar el mismo grupo de caracteres en la pantalla. Para evitar el tener que volver a escribir las mismas líneas (y usar demasiada memoria), puede bifurcarse a secciones "muy usadas" del programa GOTO. De todas formas, el uso descuidado de GOTO puede transformar los programas en laberintos confusos imposibles de comprender o depurar.

Un programa escrito metódicamente en bloques o módulos es el más fácil de analizar y depurar, puesto que cada módulo puede probarse independientemente de los otros, si se da el caso. Si examina el listado de un programa de un juego en una revista, verá que tiene un aspecto similar a:



Cómo usar una subrutina

Las subrutinas se referencian con la palabra clave GOSUB. GOSUB permite bifurcar a una subrutina y devolver el control al programa principal una vez ejecutada la subrutina. Esta sentencia es de la forma:

```
50 GOSUB 500
```

El programa principal se ejecuta normalmente hasta llegar a la sentencia número 50, que obliga al ordenador a saltar a la subrutina que comienza en la línea 500. Una vez completada la rutina, vuelve a la línea 60 del programa principal —la línea siguiente a la última ejecutada del programa principal— la subrutina debe acabar con la palabra clave RETURN. Sin ella, el ordenador no volvería al punto adecuado del programa principal.

GOSUB puede usarse en casi todos los programas en los que haya que repetir un bloque de operaciones. El siguiente programa construye una tabla de conversión de temperaturas usando tres tipos de medidas, grados centígrados, Fahrenheit y Kelvin. La subrutina de la línea 80 hace que el ordenador imprima una línea, produzca un pequeño pitido y vuelva a la línea 60. La orden STOP de la línea 70 impide que la ejecución del programa continúe en la subrutina. Si se omite STOP, el programa continúa sin llegar a RETURN de la subrutina, produciendo a continuación un mensaje de error, indicando que el sistema ha hallado un RETURN sin un GOSUB previo. Observará que la subrutina del listado siguiente está en un bucle FOR...NEXT, luego la llamará un cierto número de veces. La tabla obtenida al ejecutar el programa se muestra en la segunda pantalla de la figura:

PROGRAMA DE CONVERSION DE TEMPERATURA

```
  10 BORDER 1: PAPER 2: INK 7: C
LS
  20 PRINT AT 1,4;"C";AT 1,15;"F
";AT 1,26;"K"
  30 PRINT "--------------------
-------------"
  40 FOR c=-30 TO 140 STEP 10
  50 GO SUB 80
  60 NEXT c
  70 STOP
  80 PRINT TAB 3;c;TAB 14;(c*9/5
)+32;TAB 25;c+273
  90 BEEP .05,40
 100 PAUSE 25
 110 RETURN


0 OK, 0:1
```

```
   C              F              K
-------------------------------------
 -30            -22            243
 -20             -4            253
 -10             14            263
   0             32            273
  10             50            283
  20             68            293
  30             86            303
  40            104            313
  50            122            323
  60            140            333
  70            158            343
  80            176            353
  90            194            363
 100            212            373
 110            230            383
 120            248            393
 130            266            403
 140            284            413

9 STOP statement, 70:1
```

En este programa la subrutina no ahorra espacio alguno. Pero si se completase el programa para efectuar otros cálculos, la subrutina podría llamarse tantas veces como se desease —ahorrando tanto espacio como memoria.

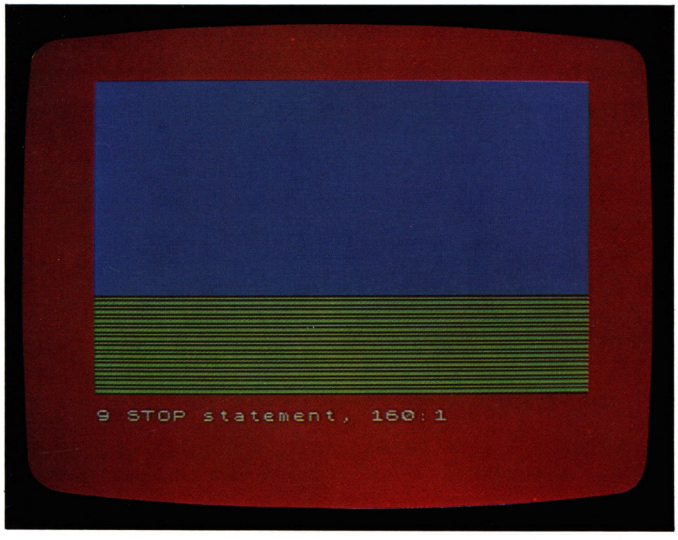
Selección de imágenes con GOSUB

En muchos programas se ofrece inicialmente un menú o repertorio de opciones a seleccionar. Cada opción se programa usualmente con GOSUB. Una vez hecha esta elección, el programa se dirige a la subrutina adecuada y exhibe la imagen elegida.

A continuación se muestra un listado que indica cómo ha de hacerse. El programa puede exhibir una de dos imágenes diferentes. Una de ellas se muestra en la pantalla de la figura siguiente. Los colores de cada imagen los produce una subrutina —la subrutina en cuestión dependerá de las entradas introducidas en la línea 20. Si se usase esta rutina en un programa real de juegos, podría usarse la siguiente elección de colores:

PROGRAMA MENU

```
  10 BORDER 1
  20 INPUT TAB 7;"DISPLAY 1 OR 2
?";a
  30 IF a=1 THEN GO SUB 300
  40 IF a=2 THEN GO SUB 400
  50 BORDER v: PAPER p: CLS
  60 PAPER q
  70 FOR r=15 TO 21
  80 FOR c=0 TO 31
  90 PRINT AT r,c;" "
 100 NEXT c: NEXT r
 110 INK 0
 120 FOR y=0 TO 55 STEP 2.5
 130 PLOT 0,y
 140 DRAW 255,0
 150 NEXT y
 160 STOP
 300 LET p=3: LET q=2: LET v=6:
RETURN
 400 LET p=1: LET q=4: LET v=2:
RETURN

0 OK, 0:1
```



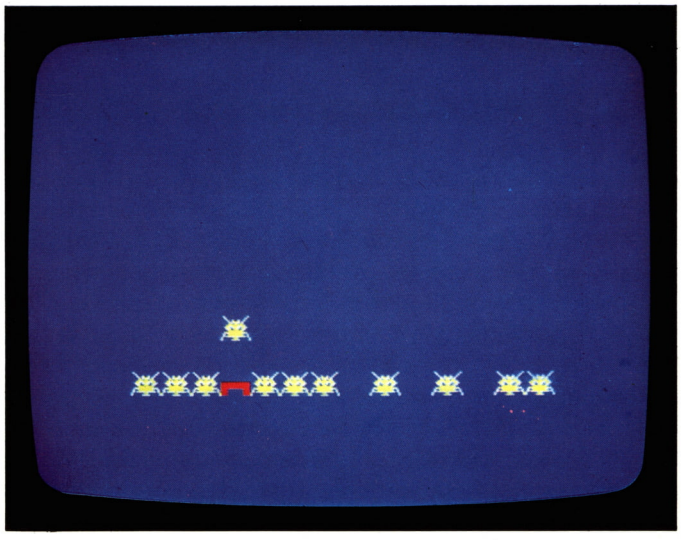
Uso de GOSUB con movimiento

El programa siguiente imprime un blanco sobre el suelo y un conjunto de extraterrestres cayendo desde puntos aleatorios del extremo superior de la pantalla. Si uno de los extraterrestres da en el blanco, la línea 240 cede el control a la subrutina de la línea 270. Esta genera una secuencia de marcos de colores y emite una serie de pitidos. Luego aparece el blanco y comienza de nuevo la ejecución:

MOVIMIENTO CON **GOSUB**

```
  10 DATA 0,0,64,32,18,11,13,31
  20 DATA 0,0,2,4,72,208,176,248
  30 DATA 59,25,15,63,79,95,129,
131
  40 DATA 220,152,240,252,242,25
0,129,193
  50 FOR n=0 TO 7: READ p
  60 POKE USR "a"+n,p
  70 NEXT n
  80 FOR n=0 TO 7: READ q
  90 POKE USR "b"+n,q
 100 NEXT n
 110 FOR n=0 TO 7: READ r
 120 POKE USR "c"+n,r
 130 NEXT n
 140 FOR n=0 TO 7: READ s
 150 POKE USR "d"+n,s
 160 NEXT n
 170 BORDER 1: PAPER 1: CLS
 180 PRINT INK 2;AT 21,8;"▄"
 190 LET x=2*INT (RND*15)
 200 FOR f=0 TO 19

scroll?
```

```
210 PRINT AT f,x;"  "
220 PRINT INK 6;AT f+1,x;"AB"
230 PRINT INK 6;AT f+2,x;"CD"
240 IF f=18 AND x=8 THEN GO SUB
270
250 NEXT f
260 GO TO 180
270 FOR t=1 TO 30
280 BEEP .005,30+t
290 FOR g=0 TO 7
300 BORDER g: NEXT g
310 NEXT t
320 BORDER 1: LET f=19
330 CLS : RETURN



0 OK, 0:1
```



TRUCOS E IDEAS

Durante la etapa de aprendizaje de manejo del Spectrum, se tropieza con numerosas técnicas de mejora de los programas basadas en el método de prueba y error. Hay, sin embargo, varias formas de ahorrar tiempo o de clasificar problemas, que si bien son sencillas y efectivas, no son en absoluto evidentes. En estas páginas se ofrecen algunos trucos que ayudarán a producir programas bien estructurados y libres de fallos.

Uso de REM como máscara o marca

Dado que el ordenador ignora todo lo que sigue a la palabra REM, puede usarse para etiquetar y probar las partes de un programa. En la página 18 se vio que REM puede usarse en la primera línea de un programa para mostrar lo que hace el programa, y cuanto más largo es el programa, más útil es el uso de REM.

Pero cuando el programa es realmente largo, es más difícil distinguir las líneas REM de las otras. Pueden destacarse poniendo unos símbolos acordados detrás de la palabra REM. Aquí tiene una forma de hacerlo.

MARCANDO LAS LINEAS **REM**

```
  10 REM ************************
  20 REM COLOUR PYRAMID PROGRAM
  30 REM © Ian Graham 1984
  40 REM ************************
  50 BORDER 0: PAPER 1: CLS
  60 DATA 144,9,127,254,73,146,6
3,252
  70 DATA 39,228,63,252,15,240,3
1,248
  80 DATA 31,248,15,240,63,252,3
9,228
  90 DATA 63,252,73,146,127,254,
144,9
 100 FOR n=0 TO 7
 110 READ p,q,r,s
 120 POKE USR "a"+n,p
 130 POKE USR "b"+n,q
 140 POKE USR "c"+n,r
 150 POKE USR "d"+n,s
 160 NEXT n
 170 FOR r=15 TO 21
 180 FOR c=0 TO 31

scroll?
```

Al leer este programa, las líneas REM destacan a primer golpe de vista.

REM es también aplicable en el desarrollo de programas. Muchas veces querrá probarse un programa para ver qué sucede al suprimir ciertas líneas. Se puede saltar parte de un programa con un GOTO o con un RUN seguido del número de una sentencia —pero esto no funciona si lo que se desea es saltar un grupo de líneas del centro. Este problema puede resolverse sin necesidad de suprimir las líneas; basta con insertar una línea REM delante de cada sentencia que se desea eludir. Esto enmascara o "desactiva" las líneas, ya que el ordenador ignora todas las órdenes que siguen a cada REM. Aquí tiene un programa en el que se ha hecho esto.

DESACTIVACION DE UNA LINEA CON **REM**

```
  10 BORDER 1: PAPER 2: INK 7: C
LS
  20 PRINT AT 1,4;"C";AT 1,15;"F
";AT 1,26;"K"
  30 PRINT "--------------------
------------"
  40 FOR c=-30 TO 140 STEP 10
  50 GO SUB 80
  60 NEXT c
  70 STOP
  80 PRINT TAB 3;c;TAB 14;(c*9/5
)+32;TAB 25;c+273
  90 REM BEEP 0.05,40
 100 PAUSE 25
 110 RETURN



0 OK, 0:1
```

Cómo comprobar bucles anidados

Es fácil complicar los bucles en un programa de forma que este no produzca los resultados deseados. Hay una forma sencilla de comprobar que los bucles están correctamente anidados, es decir, que están debidamente acoplados, sin solapamiento.

Si se anota o imprime el programa, puede enlazarse el principio y el final de cada bucle:

CONECTAR BUCLES

```
  10 BORDER 0: PAPER 0: CLS
┌─20 FOR a=0 TO 20
│ ┌30 FOR c=0 TO 7
│ │40 PRINT INK c;"■";
│ │50 PAUSE 10
│ └60 NEXT c
│ ┌70 FOR c=0 TO 7
│ │80 PRINT INK c;"■";
│ │90 PAUSE 10
│ └100 NEXT c
└─110 NEXT a



0 OK, 0:1
```

Otra alternativa es anotar todas las sentencias FOR...NEXT en un papel en el orden en el que aparecen, saltándose las líneas intermedias.

Si todos los bucles están anidados correctamente no se dará ningún cruce entre las líneas. Si se cruzan, es que los niveles de nido son incorrectos y el programa no funcionará correctamente.

Mejora de la realimentación del teclado

Cada vez que se acciona una tecla, el ordenador emite un chasquido. Esto es muy útil; sin él, habría que estar permanentemente mirando la pantalla para comprobar que el ordenador ha registrado la tecla accionada. Aunque el sonido es audible cuando se trabaja en silencio, puede no serlo a causa de ruidos procedentes de otras fuentes, haciendo difícil la programación. Si el chasquido no es lo suficientemente alto, puede aumentarse su volumen, tecleando:

```
POKE 23609,n
```

Donde n es un número entero comprendido entre 0 y 255. Cero produce un chasquido y 255 una nota larga y aguda.

Cómo editar con rapidez

Si se desea editar una línea situada en la mitad de un programa largo, puede teclearse la siguiente sentencia:

```
LIST 250
```

Esto ubica el indicador de línea en la línea que se va a editar. Pero si el listado continúa, aparece "scroll?" en la parte inferior de la pantalla. Si Vd. pulsa ahora la tecla EDIT, el programa se enrollará hacia adelante, a la página siguiente. Puede accionarse la tecla N para evitar que el listado se "enrolle", pero existe otra alternativa.

Si desea editar la línea 250 de un programa largo, teclee:

```
249
```

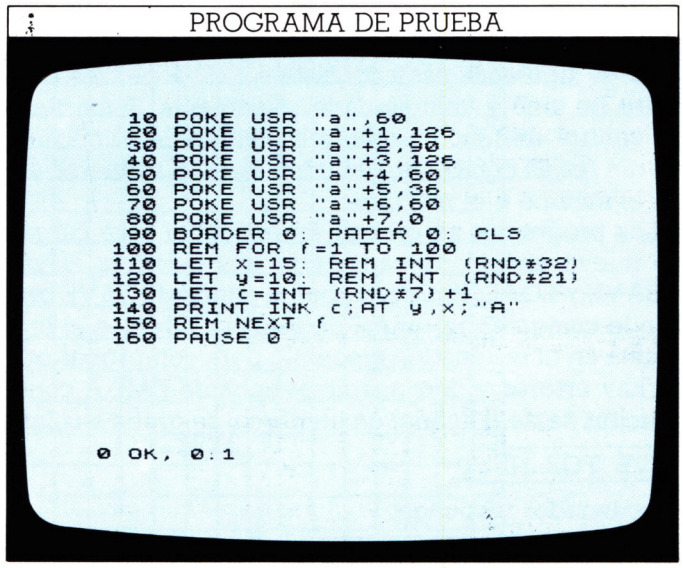
seguido de ENTER. Aparentemente no sucede nada. El indicador de línea no se mueve, pero tampoco aparece "scroll?". Si ahora acciona EDIT, la línea 250 aparece en el borde inferior de la pantalla. El 249 —o el número elegido— debe ser el número inmediatamente anterior al de la línea que se quiere editar. Hay que asegurarse que este número no es pues el número de una de las líneas del programa, pues en tal caso, al teclear el número y luego ENTER se borrará la línea de la memoria del ordenador.

Técnicas útiles de depuración

Aunque el Spectrum dispone de un gran repertorio de mensajes de error, muchas veces un programa se ejecuta erróneamente sin producir ningún mensaje. ¿Cómo se llega entonces a la fuente del problema?

Ya se ha visto que puede usarse la palabra clave REM para enmascarar partes de un programa o también pueden enlazarse los extremos de los bucles para comprobar que están debidamente anidados. Pero si esto no sirve de ayuda puede seguirse la pista al problema asignando a cada variable un valor fijo.

Sea un programa gráfico que usa la palabra clave RND para generar una imagen construida a partir de bucles. Si no funciona como se esperaba, puede sustituirse la sentencia RND por un número. Así puede calcularse el efecto que produce un número conocido al ejecutar el programa. Ahora se eliminan las líneas que marcan el principio y el final del bucle (puede usar REM para este fin). Si el resultado de una única ejecución no es tal y como se predijo, la imagen de la pantalla debe dar alguna idea sobre dónde falla el programa:



El programa de la figura es el de gráficos aleatorios de la página 49, modificado de forma que las variables aleatorias de las líneas 110-120 sean fijas. Se conservan las líneas originales, pero están desactivadas por sentencias REM. El bucle entre las líneas 100 y 150 está también desactivado por dos REMs, de manera que el programa imprime una única vez.

Si se ejecuta el programa, puede comprobarse si producen o no los resultados esperados. Si no lo hace, se ha conseguido al menos una forma más sencilla de seguir la pista a la fuente del problema.

Esta técnica puede utilizarse en cualquier programa que use variables.

Finalmente, no debe olvidarse la utilidad de la tecla BREAK para saber en qué parte del programa trabaja el ordenador. Si se ejecuta un programa que aparentemente no hace nada o que se atasca en algún punto, la tecla BREAK indica dónde se ha producido la parada. Si se lista entonces el programa, se identificará, en la mayoría de los casos, el error con la línea marcada por BREAK y podrá corregirse.

COMO GUARDAR LOS PROGRAMAS

Todo lo que Vd. teclea en el Spectrum se almacena en la memoria del ordenador siempre que éste esté conectado a la fuente de alimentación. Cuando se desconecta el ordenador, los programas desaparecen. Evidentemente, no pueden teclearse todos los programas cada vez que se conecta el ordenador. Afortunadamente, el lector dispone con seguridad de una forma barata y fácil de almacenar programas —un magnetófono corriente. El Spectrum dispone de dos enchufes en el panel posterior, marcados EAR y MIC, que deben conectarse a los enchufes correspondientes de la grabadora.

Disposición de los controles del magnetófono

Una vez que se ha conectado el ordenador a la grabadora, el siguiente paso consiste en disponer los controles de tono y volumen adecuadamente. Si existiese un control de tono, habría que disponerlo al máximo de graves. El control de volumen debe disponerse entre el mínimo y el máximo.

Los programas se graban en una cinta y se cargan de nuevo en el ordenador con dos palabras claves —SAVE y LOAD. Puede probar el comando SAVE grabando cualquier programa de este libro. Teclee el programa en el ordenador; ejecútelo para comprobar que no hay errores y desconecte el enchufe EAR. A continuación, se da al fichero un nombre y se graba en cinta:

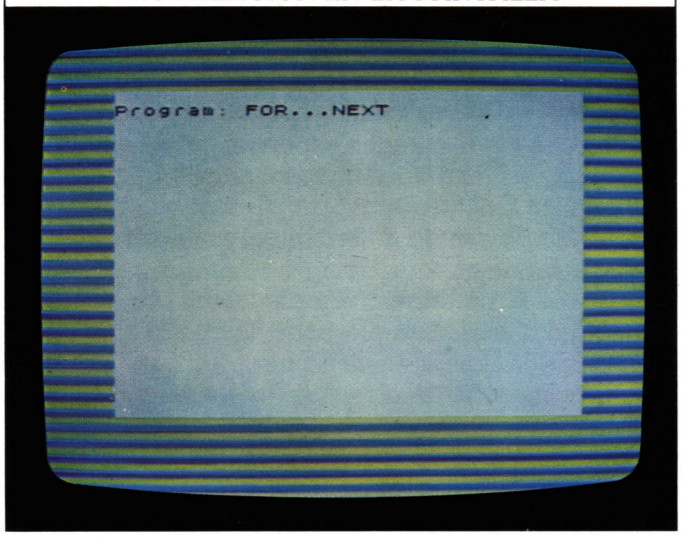
```
SAVE "FOR...NEXT"
```

El ordenador responde:

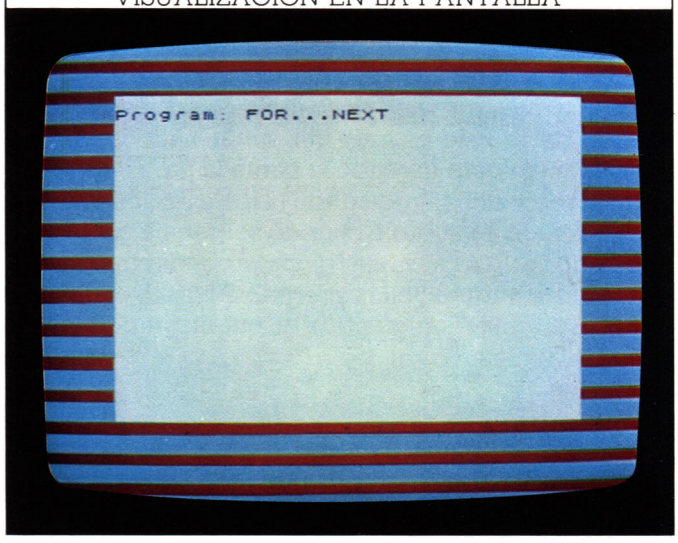
```
Start tape then press any key
```

Accionar las teclas RECORD y PLAY del magnetófono y una de las teclas del ordenador. Aparecen dos configuraciones gráficas en la pantalla:

VISUALIZACION EN LA PANTALLA



VISUALIZACIÓN EN LA PANTALLA



Una vez que se ha grabado el programa, desaparecen las líneas impresas y surge el mensaje OK. Detenga la grabadora.

Comprobación de una grabación

Para comprobar que la grabación de un programa es correcta, reconecte el cable EAR y teclee:

```
VERIFY "FOR...NEXT"
```

Accione de nuevo el botón PLAY de la grabadora. A medida que un programa grabado se almacena en el ordenador, debe aparecer la configuración de líneas azules y rojas en la pantalla, junto con el nombre del programa. En caso contrario, el programa no se ha grabado correctamente. Rebobine la cinta, suba el volumen y accione PLAY. Si aún no se obtiene el programa, interrumpa la sentencia VERIFY accionando BREAK. Intente grabar el programa de nuevo con un volumen más alto (debe desconectar el enchufe EAR).

VERIFY puede usarse para catalogar los programas de una cinta. Teclee VERIFY y "gatos" (o cualquier otro nombre que aún no haya sido usado). A medida que el ordenador lee un programa de la cinta, el nombre del fichero correspondiente aparece en la pantalla.

Almacenar un programa en la memoria del ordenador

Vd. desea almacenar un programa en la memoria del ordenador. Teclee:

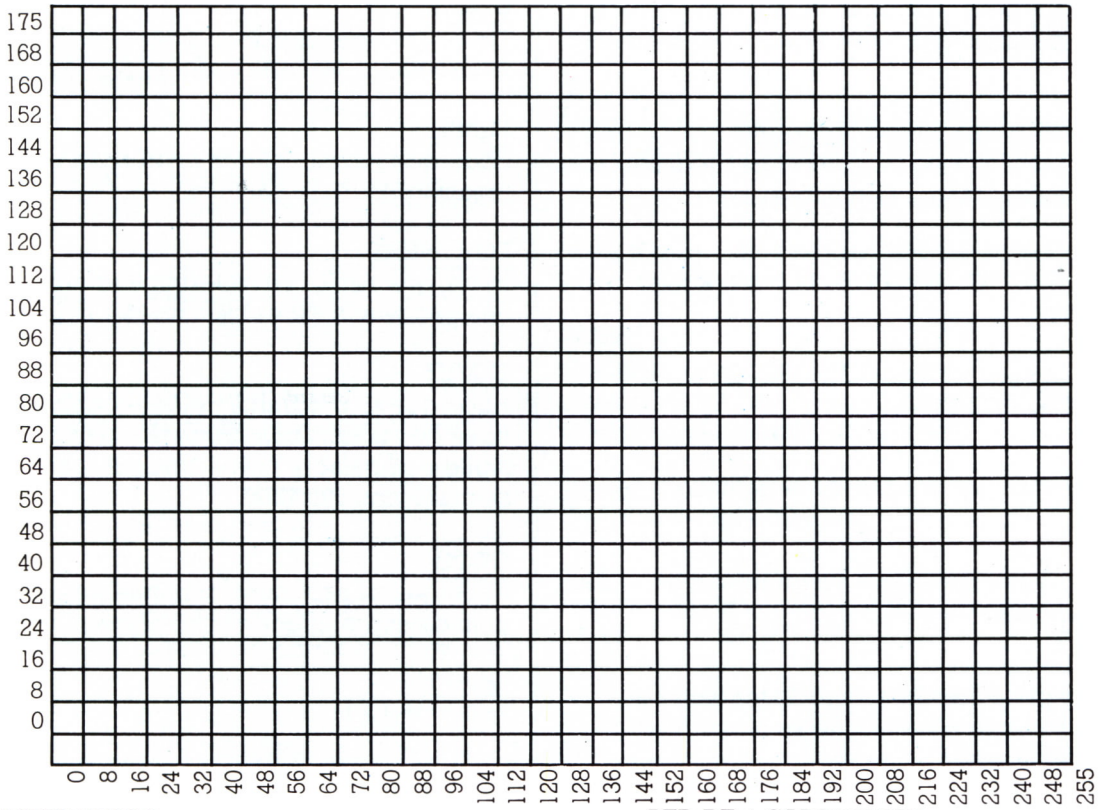
```
LOAD "FOR...NEXT"
```

La cinta debe estar rebobinada. Cuando el ordenador se encuentra en el programa, exhibe "program" seguido del nombre del fichero en la pantalla. Una vez que el programa reside en la memoria, aparece el mensaje OK. Ya puede proceder a la ejecución.

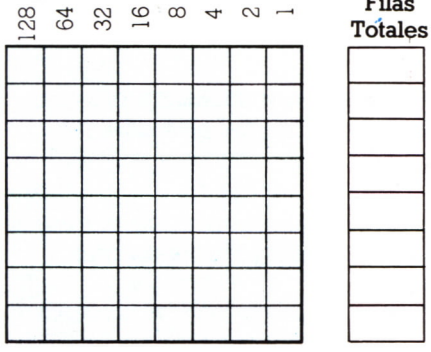
REDES DE GRAFICOS Y CARACTERES

La red de la figura muestra las coordenadas de los puntos de la pantalla cuando se utilizan órdenes de tipo gráfico. Un punto de la pantalla se identifica mediante dos coordenadas x,y. La primera fija la posición horizontal, que se mide desde el borde izquierdo de la pantalla; la segunda determina la posición vertical medida desde el borde inferior. Cualquier carácter impreso ocupa un área de 8 unidades gráficas de ancho y 8 de alto. No se puede imprimir en las dos líneas inferiores de la pantalla.

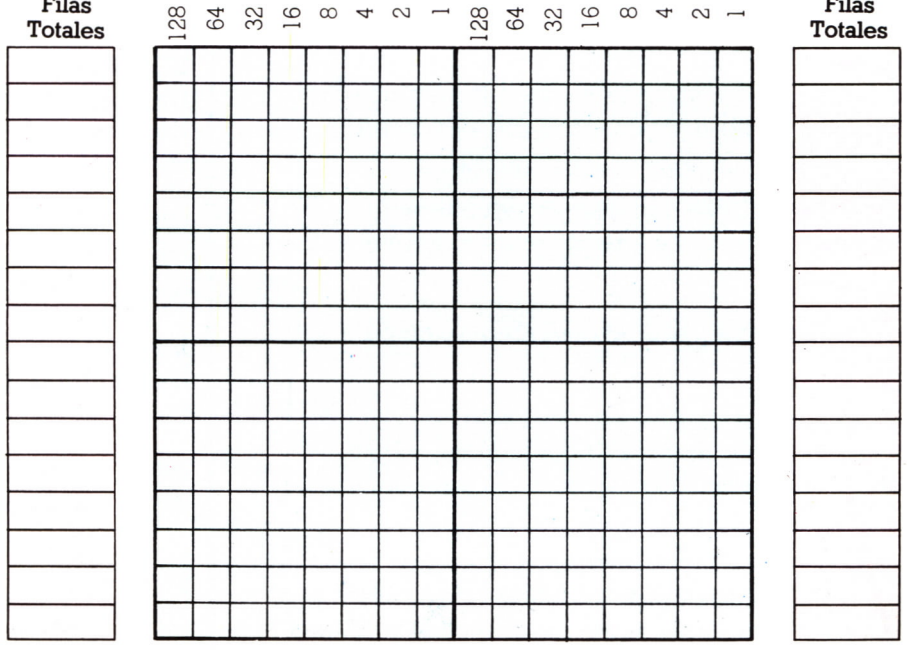
RED DE COORDENADAS DE GRAFICOS



RED DE CARACTER UNICO



RED DE 4 CARACTERES



Redes de caracteres Vd. puede usarlas para diseñar un carácter único (arriba), o un símbolo de hasta 8 caracteres (derecha). Puede Vd. dibujar con lápiz las figuras deseadas en las redes y luego sumar los totales en las columnas de valores totales. Estos valores se usan en los programas con las palabras clave POKE USR. Las teclas A-U están disponibles para la programación de caracteres definidos por el usuario.

GLOSARIO

Las palabras en negrita son palabras clave del BASIC.

AT
Se utiliza con **PRINT** para ubicar caracteres en la pantalla.

BASIC
Beginners' All-purpose Symbolic Instruction Code; el lenguaje de programación de alto nivel más utilizado.

BEEP
Hace que el ordenador genere un pitido corto o una nota, cuya duración y tono se determina mediante los números que acompañan a la palabra clave.

BIN
Transforma un número binario en el equivalente número decimal.

Bit
Un dígito binario —0 ó 1.

BORDER
Modifica el color de los bordes de la pantalla.

BRIGHT
Imprime caracteres específicos en un tono más brillante de su color.

Bucle
Secuencia de sentencias de un programa que se ejecuta repetidamente o hasta que se alcanza la condición especificada.

Byte
Un conjunto de ocho bits.

Cadena
Secuencia de caracteres tratada como un objeto único —el nombre de una persona, por ejemplo.

Chip
Paquete único que contiene un circuito electrónico completo. Se denomina también circuito integrado.

CLS
Borra el área dedicada a texto de la pantalla.

CPU
Unidad Central de Proceso. Está normalmente contenida en un único *chip*. Es la encargada de realizar las operaciones aritméticas y de control del sistema.

Cursor
Símbolo parpadeante de la pantalla que señala el lugar en el que debe aparecer el carácter siguiente.

DATA
Almacena una línea de datos. El ordenador trata todo lo que sigue a la palabra clave **DATA** como información para utilizar eventualmente en el programa. Se utiliza junto con **READ**.

Diagrama de flujo
Representación de los pasos necesarios en la resolución de un problema.

DRAW
Traza una línea desde el origen gráfico 0,0 o desde el último punto visitado hasta el punto especificado.

FLASH
Hace que los caracteres aparezcan y desaparezcan fugazmente de la pantalla.

FOR...NEXT
Bucle que repite un determinado número de veces una secuencia de sentencias de un programa.

GOSUB
Hace que el programa ceda el control a la primera línea de la subrutina ubicada a partir del número que sigue a la palabra clave. Una subrutina debe acabar siempre con **RETURN.**

GOTO
Hace que el programa continúe a partir de la línea titulada con el número que sigue a la palabra clave.

GRAPH
Tecla de 10-11.

Hardware
Maquinaria física de un ordenador, en contraposición a los programas (software).

IF...THEN
Hace que el ordenador realice unas acciones determinadas cuando se cumple la condición impuesta.

INK
Modifica el color de los gráficos y del texto que aparece en la pantalla.

INPUT
Indica al ordenador que debe esperar datos procedentes del exterior.

INT
Transforma un número decimal en entero.

Interface
Conexión física y lógica situada entre el ordenador y otro dispositivo del equipo.

INVERSE
Conmuta el color del fondo al del texto (INK) y viceversa.

K
Abreviatura de kilobyte (1.024 bytes).

LET
Asigna un valor a una variable.

LIST
Hace que el ordenador exhiba el programa almacenado en su memoria.

LOAD
Realiza la transferencia de un programa desde la cinta de un *cassette* hasta la memoria del ordenador.

NEW
Elimina un programa de la memoria del ordenador.

OVER
Permite imprimir nuevos caracteres sobre caracteres ya existentes sin necesidad de borrar los anteriores.

PAPER
Cambia el color del fondo de la pantalla.

PAUSE
Detiene el programa durante el lapso de tiempo indicado por un número expresado en cincuentavos de segundo.

PLOT
Dibuja un único punto en la pantalla en el lugar especificado por las coordenadas que siguen a la palabra clave.

POKE USR
Almacena un número que reprograma una tecla para producir caracteres definidos por el usuario.

PRINT
Imprime lo que sigue en la pantalla.

Ram
Memoria de Acceso Aleatorio (memoria volátil). El contenido de esta memoria desaparece cuando se desconecta la fuente de alimentación. Ver Rom.

READ
Informa al ordenador que debe tomar un número determinado de datos de una sentencia DATA, de forma que puedan usarse en un programa.,

RESET
Botón de 8.

REM
Permite al usuario introducir comentarios en un programa. El ordenador ignora todo lo que sigue a la palabra REM.

RESTORE
Reinicializa los datos de una sentencia DATA, de forma que éstos puedan usarse más de una vez en un programa.

RETURN
Marca el final de una subrutina (ver GOSUB).

RND
Genera un número al azar entre 0 y 1 que puede utilizarse para generar secuencias no predecibles.

Rom
Memoria de Sólo Lectura. programada por el fabricante, y que sólo puede leerlo el usuario.

SAVE
Graba un programa residente en la memoria del ordenador en una cinta. El programa se identifica mediante el nombre del fichero.

Software
Programas del ordenador.

SQR
Produce la raíz cuadrada del número que lo sigue.

STEP
Marca el incremento o decremento de un bucle FOR...NEXT.

STOP
Detiene un programa e imprime el número de la línea en la que aparece.

Subrutina
Una parte de un programa que puede ser llamada cuando sea necesario. Puede utilizarse, por ejemplo, para exhibir una determinada configuración en la pantalla o para realizar un conjunto de operaciones repetidamente.

TAB
Se utiliza junto con **PRINT** para determinar en qué lugar debe imprimirse una cadena de caracteres.

Variable
Posición de memoria etiquetada en la que puede almacenarse u obtenerse información.

VERIFY
Comprueba que un programa residente en memoria se ha grabado correctamente en cinta mediante la palabra clave SAVE.

INDICE

Las entradas principales van en **negrita.**

Curso Visual en Pantalla

"...para todo el que quiera llegar a ser mejor programador. El método de enseñanza es infalible"

Nigel Searle, Director
SINCLAIR RESEARCH

Original y apasionante curso de programación para los usuarios del ZX Spectrum+.

Más de 150 fotografías de pantallas con listados de programas que le muestran exactamente lo que aparecerá en su pantalla.

Otros títulos de la serie:

- **ZX Spectrum,**
- **Apple IIe,**
- **Commodore 64**

MicroTextos